ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT127-3-2-PFDA

PROGRAMMING FOR DATA ANALYSIS

APD2F2209CS(CA)

HAND OUT DATE: 10 OCTOBER 2022

HAND IN DATE: 28 NOVEMBER 2022

WEIGHTAGE: 50%

INSTRUCTIONS TO CANDIDATES:

1 **Submit your assignment at the administrative counter.**

2 **Students are advised to underpin their answers with the use of references (cited using the American Psychological Association (APA) Referencing).**

3 **Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**

4 **Cases of plagiarism will be penalized.**

5 **The assignment should be bound in an appropriate style (comb bound or stapled).**

6 **Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.**

7 **You must obtain 50% overall to pass this module.**

Table of Contents

1.0 Introduction

For decades, house rent tends to be essential to determine while purchasing a house. There are a few aspects that a tenant should look out for while making their decisions, which includes BHK, rent price, unit size, floor number, type of area, locality, city located, status of furnishings, preferred tenants, number of bathrooms and the contact person. To ensure that the buyer's needs are satisfied from a house owners' perspective, they should grasp an understanding on the preferences of their renting units according to diverse demographics such as bachelors and family. The geographical factors have also played a big role in tenants and housing developers' decision in either renting or building a house, specifically in the total number of floors present for each building as it can be influenced by several characteristics in geography. Tenants are usually attracted to practical and reasonable renting prices. Hence, the factors influencing the property pricing should be studied in an accurate manner for clear decision making.

A dataset is present for analysis to ensure that all the above problem statements are solved through insights given from the data relating to the statements. The comparison of data can be furtherly done through executing the data exploration, cleaning, manipulation, transformation, and visualization. The process assists to reduce the risks inherent while making decisions by giving useful statistics presented in graphs and charts that are easily understood by key decision makers. By performing data analysis, resources for the business will not go to waste as it provides a direction for the company to focus on relevant demographic groups for advertising efforts. For this analysis, the "House_Rent_Dataset.csv" dataset is chosen to generate several outcomes through analysis.

2.0 Assumptions

General:

1. The dataset can be analysed from the tenants' or house owners' perspective by using the same columns and rows.
2. The columns of the dataset are renamed as the following:

Original Column Name	**New Column name**
Posted.On	Posting Date
BHK	BHK In Unit
Rent	Rent Price
Size	Unit Size
Floor	Floor Number
Area.Type	Type of Area
Area.Locality	Locality
City	City Located
Furnishing.Status	Furnishings
Tenant.Preferred	Preferred Tenants
Bathroom	Number of Bathroom
Point.of.Contact	Contact Person

Table 2-1: New Column Names

3. The bachelor/family data present in the "Preferred Tenants" column impacts both the bachelor and family while data analyzation.
4. The posting date should be in the format YYYY-MM-DD.
5. The unit size is measured in square feet.
6. The rent price is measured in Indian Rupee.
7. Extreme outliers should be replaced with median as the median has lower influence by the outliers and skewed data comparing with the mean.
8. The dataset is present with the .csv format.

Tenants:

1. Buyers can look into the factors affecting the pricing of the units.

House Owner:

1. The owner is interested in knowing the preferences of bachelors and families while choosing a house.
2. The owner wants to know the demand of property rental in each city and the factors influencing it.
3. The owner wishes to have insights on how the locality and city affects the number of floors of one building.
4. The owner wants to analyze the behavior of people in searching for houses according to month and location.

3.0 Data Pre-processing

3.1 Data Import

Code:

```
#Section 1: Data Import
#Import the data set
house_data = read.csv("C:\\Users\\chery\\Desktop\\Semester 1\\PDFA\\House_Rent_Dataset.csv", header=TRUE)
#View data in table form
View(house_data)
```

Figure 3-1: Import Dataset

Output:

	Posted.On	BHK	Rent	Size	Floor	Area.Type	Area.Locality	City	Furnishing.Status	Tenant.Preferred	Bathroom	Point.of.Contact
1	5/18/2022	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
2	5/13/2022	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
3	5/16/2022	2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
4	7/4/2022	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
5	5/9/2022	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished	Bachelors	1	Contact Owner
6	4/29/2022	2	7000	600	Ground out of 1	Super Area	Thakurpukur	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
7	6/21/2022	2	10000	700	Ground out of 4	Super Area	Malancha	Kolkata	Unfurnished	Bachelors	2	Contact Agent
8	6/21/2022	1	5000	250	1 out of 2	Super Area	Malancha	Kolkata	Unfurnished	Bachelors	1	Contact Agent
9	6/7/2022	2	26000	800	1 out of 2	Carpet Area	Palm Avenue Kolkata, Ballygunge	Kolkata	Unfurnished	Bachelors	2	Contact Agent
10	6/20/2022	2	10000	1000	1 out of 3	Carpet Area	Natunhat	Kolkata	Semi-Furnished	Bachelors/Family	2	Contact Owner
11	5/23/2022	3	25000	1200	1 out of 4	Carpet Area	Action Area 1, Rajarhat Newtown	Kolkata	Semi-Furnished	Bachelors/Family	2	Contact Agent
12	6/7/2022	1	5000	400	1 out of 1	Carpet Area	Keshtopur	Kolkata	Unfurnished	Bachelors/Family	1	Contact Agent
13	5/14/2022	1	6500	250	1 out of 4	Carpet Area	Tarulia, Keshtopur	Kolkata	Furnished	Bachelors	1	Contact Owner
14	5/9/2022	1	5500	375	1 out of 2	Carpet Area	Dum Dum Metro	Kolkata	Unfurnished	Bachelors/Family	1	Contact Agent
15	5/5/2022	3	8500	900	Ground out of 2	Carpet Area	Paschim Barisha	Kolkata	Unfurnished	Bachelors	2	Contact Owner
16	6/1/2022	3	40000	1286	1 out of 1	Carpet Area	New Town Action Area 1	Kolkata	Furnished	Bachelors/Family	2	Contact Owner
17	5/17/2022	2	6000	600	1 out of 2	Super Area	Barasat	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
18	6/20/2022	2	10000	800	Ground out of 2	Super Area	Behala	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
19	6/9/2022	2	11000	2000	Ground out of 3	Carpet Area	Behala Chowrasta	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
20	6/9/2022	2	6000	660	1 out of 2	Super Area	Behala	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
21	7/2/2022	2	7900	650	1 out of 2	Carpet Area	Santoshpur	Kolkata	Unfurnished	Family	1	Contact Owner
22	6/14/2022	2	9000	400	2 out of 3	Carpet Area	Garia Station, Garia	Kolkata	Unfurnished	Bachelors	2	Contact Owner
23	6/15/2022	1	4000	300	Ground out of 4	Carpet Area	Garia Station, Garia	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
24	6/15/2022	3	6500	1600	Ground out of 2	Super Area	Joka	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
25	5/28/2022	1	8000	400	1 out of 2	Super Area	Sreebhumi	Kolkata	Semi-Furnished	Bachelors	1	Contact Agent
26	5/22/2022	2	7000	1000	1 out of 1	Super Area	Rajarhat	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner

Figure 3-2: Dataset in Table Form

The dataset “House_Rent_Dataset.csv” is read through the read.csv() function with the header present and stored in the house_data variable. Then, the house data is presented in the form of table by executing View(house_data).

3.2 Data Exploration

Before initiating data cleaning, the data is explored to determine the aspects that the data should be enhanced on. The exploration covers on determining the kind, complexity, sufficiency, missing data and the dependency of each variable (Nguyen, 2021).

1. Changing the column name

Code:

```
#Renaming the columns of the data set
names(house_data) = c("Posting Date", "BHK In Unit", "Rent Price", "Unit Size", "Floor Number", "Type of Area", "Locality",
                      "City Located", "Furnishings", "Preferred Tenants", "Number of Bathroom", "Contact Person")
house_data
```

Figure 3-3: Renaming Columns

Output:

Posting Date	BHK In Unit	Rent Price	Unit Size	Floor Number	Type of Area	Locality	City Located	Furnishings	Preferred Tenants	Number of Bathroom	Contact Person
5/18/2022	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
5/13/2022	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
5/16/2022	2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner

Figure 3-4: Renamed Columns

To provide more meaningful labels to each column, a new set of column names are defined as a vector which consists of Posting Date, BHK In Unit, Rent Price, Floor Number, Type of Area, Locality, City Located, Furnishings, Preferred Tenants, Number of Bathroom and Contact Person. The initial column names are replaced by assigning the new column names to names(house_data).

2. Calculate basic descriptive statistics

Code:

```
#Calculate basic descriptive statistics
summary(house_data)
```

Figure 3-5: Summary Function

Output:

```
> summary(house_data)
 Posting Date        BHK In Unit      Rent Price        Unit Size
 Length:4746        Min.   :1.000   Min.   :   1200   Min.   :  10.0
 Class :character   1st Qu.:2.000   1st Qu.:  10000   1st Qu.: 550.0
 Mode  :character   Median :2.000   Median :  16000   Median : 850.0
                    Mean   :2.084   Mean   :  34993   Mean   : 967.5
                    3rd Qu.:3.000   3rd Qu.:  33000   3rd Qu.:1200.0
                    Max.   :6.000   Max.   :3500000   Max.   :8000.0
 Floor Number       Type of Area         Locality
 Length:4746        Length:4746        Length:4746
 Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character



 City Located       Furnishings        Preferred Tenants
 Length:4746        Length:4746        Length:4746
 Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character



 Number of Bathroom Contact Person
 Min.   : 1.000     Length:4746
 1st Qu.: 1.000     Class :character
 Median : 2.000     Mode  :character
 Mean   : 1.966
 3rd Qu.: 2.000
 Max.   :10.000
```

Figure 3-6: Summary of Statistics

The summary() function is utilized to summarize each variable in the house data dataset. For each of the numerical variables, the output displays the min, first quartile, median, mean, third quartile and max values. As for categorical variables, the output only displays the column class and mode. Through the summary, basic insights about the data can be extracted for further exploration and cleaning.

3. Calculate the number of rows

Code:

```
#Calculate the number of rows
nrow(house_data)
```

Figure 3-7: Calculate Number of Rows

Output:

```
> #Calculate the number of rows
> nrow(house_data)
[1] 4746
```

Figure 3-8: Total Number of Rows

The number of rows in the dataset is calculated through the nrow() function. A total of 4746 rows of data is present within the dataset.

4. Calculate the number of columns

Code:

```
#Calculate the number of columns
ncol(house_data)
```

Figure 3-9: Calculate Number of Columns

Output:

```
> #Calculate the number of columns
> ncol(house_data)
[1] 12
```

Figure 3-10: Total Number of Columns

The number of columns in the dataset is calculated through the ncol() function. A total of 12 columns of data is present within the dataset.

5. Looking into the first 10 rows of the dataset

Code:

```
#Looking into the first 10 rows of the data set
head(house_data, 10)
```

Figure 3-11: Head Function

Output:

```
> head(house_data, 10)
   Posting Date BHK In Unit Rent Price Unit Size    Floor Number
1     5/18/2022           2      10000      1100 Ground out of 2
2     5/13/2022           2      20000       800      1 out of 3
3     5/16/2022           2      17000      1000      1 out of 3
4      7/4/2022           2      10000       800      1 out of 2
5      5/9/2022           2       7500       850      1 out of 2
6     4/29/2022           2       7000       600 Ground out of 1
7     6/21/2022           2      10000       700 Ground out of 4
8     6/21/2022           1       5000       250      1 out of 2
9      6/7/2022           2      26000       800      1 out of 2
10    6/20/2022           2      10000      1000      1 out of 3
   Type of Area                        Locality City Located
1    Super Area                          Bandel      Kolkata
2    Super Area        Phool Bagan, Kankurgachi      Kolkata
3    Super Area         Salt Lake City Sector 2      Kolkata
4    Super Area                     Dumdum Park      Kolkata
5   Carpet Area                   South Dum Dum      Kolkata
6    Super Area                     Thakurpukur      Kolkata
7    Super Area                        Malancha      Kolkata
8    Super Area                        Malancha      Kolkata
9   Carpet Area Palm Avenue Kolkata, Ballygunge      Kolkata
10  Carpet Area                       Natunhat      Kolkata
      Furnishings Preferred Tenants Number of Bathroom Contact Person
1     Unfurnished  Bachelors/Family                 2  Contact Owner
2  Semi-Furnished  Bachelors/Family                 1  Contact Owner
3  Semi-Furnished  Bachelors/Family                 1  Contact Owner
4     Unfurnished  Bachelors/Family                 1  Contact Owner
5     Unfurnished         Bachelors                 1  Contact Owner
6     Unfurnished  Bachelors/Family                 2  Contact Owner
7     Unfurnished         Bachelors                 2  Contact Agent
8     Unfurnished         Bachelors                 1  Contact Agent
9     Unfurnished         Bachelors                 2  Contact Agent
10 Semi-Furnished  Bachelors/Family                 2  Contact Owner
```

Figure 3-12: First 10 Rows of The Dataset

The first 10 rows of the dataset can be displayed through the head() function, with stating the number of rows that the function should display. The similarity of the first 10 rows lies on the city located, which is Kolkata.

6. Looking into the last 10 rows of the dataset

Code:

```
#Looking into the last 10 rows of the data set
tail(house_data, 10)
```

Figure 3-13:Tail Function

Output:

```
> tail(house_data, 10)
     Posting Date BHK In Unit Rent Price Unit Size           Floor Number Type of Area                  Locality City Located    Furnishings
4737    6/28/2022           3      15000      1500 Lower Basement out of 2   Super Area                  Almasguda    Hyderabad Semi-Furnished
4738     7/7/2022           3      15000      1500 Lower Basement out of 2   Super Area                  Almasguda    Hyderabad Semi-Furnished
4739     7/6/2022           2      17000       855             4 out of 5  Carpet Area Godavari Homes, Quthbullapur    Hyderabad    Unfurnished
4740     7/6/2022           2      25000      1040             2 out of 4  Carpet Area                  Gachibowli    Hyderabad    Unfurnished
4741     6/2/2022           2      12000      1350             2 out of 2   Super Area                   Old Alwal    Hyderabad    Unfurnished
4742    5/18/2022           2      15000      1000             3 out of 5  Carpet Area                Bandam Kommu    Hyderabad Semi-Furnished
4743    5/15/2022           3      29000      2000             1 out of 4   Super Area         Manikonda, Hyderabad    Hyderabad Semi-Furnished
4744    7/10/2022           3      35000      1750             3 out of 5  Carpet Area         Himayath Nagar, NH 7    Hyderabad Semi-Furnished
4745     7/6/2022           3      45000      1500           23 out of 34  Carpet Area                  Gachibowli    Hyderabad Semi-Furnished
4746     5/4/2022           2      15000      1000             4 out of 5  Carpet Area              Suchitra Circle    Hyderabad    Unfurnished
     Preferred Tenants Number of Bathroom Contact Person
4737            Family                 3  Contact Owner
4738  Bachelors/Family                 3  Contact Owner
4739         Bachelors                 2  Contact Agent
4740         Bachelors                 2  Contact Owner
4741  Bachelors/Family                 2  Contact Owner
4742  Bachelors/Family                 2  Contact Owner
4743  Bachelors/Family                 3  Contact Owner
4744  Bachelors/Family                 3  Contact Agent
4745            Family                 2  Contact Agent
4746         Bachelors                 2  Contact Owner
```

Figure 3-14: Last 10 Rows of The Dataset

The last 10 rows of the dataset can be displayed through the tail() function, with stating the number of rows that the function should display. The similarity of the last 10 rows lies on the city located, which is Hyderabad.

7. View 10 random rows in the dataset

Code:

```
#View 10 random rows of the data set
sample_n(house_data, 10)
```

Figure 3-15: sample_n Function

Output:

```
> sample_n(house_data, 10)
   Posting Date BHK In Unit Rent Price Unit Size     Floor Number Type of Area                                Locality City Located    Furnishings
1     6/25/2022           4     300000      1600     53 out of 78  Carpet Area                  Trump Tower, Worli       Mumbai Semi-Furnished
2      5/4/2022           2      15000      1000       4 out of 5  Carpet Area                    Suchitra Circle    Hyderabad    Unfurnished
3     5/27/2022           3     100000      2000       3 out of 14 Carpet Area                           Alwarpet      Chennai Semi-Furnished
4      7/9/2022           3     170000      1150      8 out of 12  Carpet Area                        Bandra West       Mumbai Semi-Furnished
5     6/23/2022           4     850000      3200       2 out of 4  Carpet Area                       Breach Candy       Mumbai      Furnished
6     6/20/2022           3      30000      1893 Ground out of 18   Super Area                          Begur Road    Bangalore Semi-Furnished
7     5/12/2022           1      12500       450       2 out of 3  Carpet Area                    Patel Nagar West            Delhi      Furnished
8     6/30/2022           1       6500       700       1 out of 4   Super Area Electronic City Phase 2, Electronic City    Bangalore    Unfurnished
9     5/22/2022           2      20000      1180       1 out of 4   Super Area                        C V Raman Nagar    Bangalore Semi-Furnished
10    6/23/2022           2      30000       720      5 out of 14  Carpet Area            Samarth Garden, Bhandup West       Mumbai Semi-Furnished
   Preferred Tenants Number of Bathroom Contact Person
1   Bachelors/Family                 4  Contact Agent
2         Bachelors                  2  Contact Owner
3         Bachelors                  3  Contact Agent
4   Bachelors/Family                 3  Contact Agent
5   Bachelors/Family                 4  Contact Agent
6         Bachelors                  4  Contact Agent
7   Bachelors/Family                 1  Contact Agent
8         Bachelors                  1  Contact Owner
9   Bachelors/Family                 2  Contact Owner
10  Bachelors/Family                 2  Contact Agent
```

Figure 3-16: 10 Random Rows in The Dataset

Random rows of the dataset is displayed through utilizing the sample_n() function, where it is useful in determining diverse rows of data to gain a deeper comprehension on the context of the data.

8. Categorize categorical data

Code:

```
#Categorise floor number data
factor(house_data$`Floor Number`)

#Categorise type of area data
factor(house_data$`Type of Area`)

#Categorise locality data
factor(house_data$Locality)

#Categorise city located data
factor(house_data$`City Located`)

#Categorise furnishing data
factor(house_data$Furnishings)

#Categorise preferred tenants
factor(house_data$`Preferred Tenants`)

#Categorise Contact Person
factor(house_data$`Contact Person`)
```

Figure 3-17: factor Function

Output:

```
 [ reached getOption("max.print") -- omitted 3746 entries ]
480 Levels: 1 1 out of 1 1 out of 10 1 out of 11 1 out of 12 1 out of 13 1 out of 14 1 out of 15 1 out of 16 1 out of 19 1 out of 2 ... Upper Basement out of
 9
 [ reached getOption("max.print") -- omitted 3746 entries ]
Levels: Built Area Carpet Area Super Area
 [ reached getOption("max.print") -- omitted 3746 entries ]
2235 Levels:  Beeramguda, Ramachandra Puram, NH 9  in Boduppal, NH 2 2  in Erragadda, NH 9  in Miyapur, NH 9 117 Residency, Chembur East ... Zamin Pallavara
m, Pallavaram
 [ reached getOption("max.print") -- omitted 3746 entries ]
Levels: Bangalore Chennai Delhi Hyderabad Kolkata Mumbai
[1000] Semi-Furnished
 [ reached getOption("max.print") -- omitted 3746 entries ]
Levels: Furnished Semi-Furnished Unfurnished
 [ reached getOption("max.print") -- omitted 3746 entries ]
Levels: Bachelors Bachelors/Family Family
 [991] Contact Agent Contact Agent Contact Agent Contact Agent Contact Agent Contact Agent Contact Agent Contact Agent Contact Agent Contact Agent
 [ reached getOption("max.print") -- omitted 3746 entries ]
Levels: Contact Agent Contact Builder Contact Owner
```

Figure 3-18: Categorization of Categorical Data

The factor() function is utilized to categorise and store data of strings and integers as levels (Johnson D. , 2022). For this scenario, only the x value is passed into the function as the desired outcome is the basic details of the categorization. Through implementing factor(), it

can be deduced that the type of categorical variable for the Floor Number, Type of Area, Locality, City Located, Furnishings, Preferred Tenants and Contact Person is nominal as it order does not matter.

9. Count of rows according to variable

Code:

```
#Count of rows according to variable
house_data %>% count(`Type of Area`)
house_data %>% count(`BHK In Unit`)
house_data %>% count(`City Located`)
house_data %>% count(Furnishings)
house_data %>% count(`Preferred Tenants`)
house_data %>% count(`Number of Bathroom`)
```

Figure 3-19: Counting Rows According to Variable

Output:

```
> house_data %>% count(`Type of Area`)
  Type of Area    n
1   Built Area    2
2  Carpet Area 2298
3   Super Area 2446
> house_data %>% count(`BHK In Unit`)
  BHK In Unit    n
1           1 1167
2           2 2265
3           3 1098
4           4  189
5           5   19
6           6    8
> house_data %>% count(`City Located`)
  City Located   n
1    Bangalore 886
2      Chennai 891
3        Delhi 605
4    Hyderabad 868
5      Kolkata 524
6       Mumbai 972
> house_data %>% count(Furnishings)
    Furnishings    n
1      Furnished  680
2 Semi-Furnished 2251
3    Unfurnished 1815
> house_data %>% count(`Preferred Tenants`)
  Preferred Tenants    n
1         Bachelors  830
2  Bachelors/Family 3444
3            Family  472
> house_data %>% count(`Number of Bathroom`)
  Number of Bathroom    n
1                  1 1474
2                  2 2291
3                  3  749
4                  4  156
5                  5   60
6                  6   12
7                  7    3
8                 10    1
```

Figure 3-20: The Total Count of Each Column

The count() function from the dplyr package enables the counting of unique values of one or more variables (dplyr, n.d.). The total amount of values presents within the column selected is broken down according to the values available within the column. The categorical columns of the dataset are placed within the count() function to obtain insights on the total amount of values within the Type of Area, BHK In Unit, City Located, Furnishings, Preferred Tenants and Number of Bathroom columns. It can be deduced that the values in type of area, BHK In Unit and Number of Bathroom has a vast count range.

10. List structure of data

Code:

```
#List structure of data (including the data types)
str(house_data)
```

Figure 3-21: Obtaining Data List Structure

Output:

```
> str(house_data)
'data.frame':   4746 obs. of  12 variables:
 $ Posting Date      : chr  "5/18/2022" "5/13/2022" "5/16/2022" "7/4/2022" ...
 $ BHK In Unit       : int  2 2 2 2 2 2 2 1 2 2 ...
 $ Rent Price        : int  10000 20000 17000 10000 7500 7000 10000 5000 26000 10000 ...
 $ Unit Size         : int  1100 800 1000 800 850 600 700 250 800 1000 ...
 $ Floor Number      : chr  "Ground out of 2" "1 out of 3" "1 out of 3" "1 out of 2" ...
 $ Type of Area      : chr  "Super Area" "Super Area" "Super Area" "Super Area" ...
 $ Locality          : chr  "Bandel" "Phool Bagan, Kankurgachi" "Salt Lake City Sector 2" "Dumdum Park" ...
 $ City Located      : chr  "Kolkata" "Kolkata" "Kolkata" "Kolkata" ...
 $ Furnishings       : chr  "Unfurnished" "Semi-Furnished" "Semi-Furnished" "Unfurnished" ...
 $ Preferred Tenants : chr  "Bachelors/Family" "Bachelors/Family" "Bachelors/Family" "Bachelors/Family" ...
 $ Number of Bathroom: int  2 1 1 1 1 2 2 1 2 2 ...
 $ Contact Person    : chr  "Contact Owner" "Contact Owner" "Contact Owner" "Contact Owner" ...
>
```

Figure 3-22: Data List Structure

The str() function is mainly used for displaying the structure of the objects internally with more compactness. It states that the data frame has 4746 observations (rows) of 12 variables (columns) and it describes the data type of each columns. The function is deemed useful to reveal issues on column naming and class errors. The type of data for each column are as follows:

Column Name	**Data Type**
Posting Date	Character
BHK In Unit	Integer

Rent Price	Integer
Unit Size	Integer
Floor Number	Character
Type of Area	Character
Locality	Character
City Located	Character
Furnishings	Character
Preferred Tenants	Character
Number of Bathrooms	Integer
Contact Person	Character

Table 3-1: Data Types of Columns

According to the information, it can be concluded that the data type of the Posting Date column is inaccurate as it is in the character format. The date format seems to be a more accurate data type representation for the column.

11. Looking into unique values

Code:

```
#Looking into unique values
unique(house_data$Furnishings)
unique(house_data$`Preferred Tenants`)
unique(house_data$`City Located`)
unique(house_data$`Type of Area`)
unique(house_data$`Contact Person`)
```

Figure 3-23: Determining Unique Values

Output:

```
> unique(house_data$Furnishings)
[1] "unfurnished"    "semi-furnished" "furnished"
> unique(house_data$`Preferred Tenants`)
[1] "bachelors/family" "bachelors"        "family"
> unique(house_data$`City Located`)
[1] "kolkata"   "mumbai"    "bangalore" "delhi"     "chennai"
[6] "hyderabad"
> unique(house_data$`Type of Area`)
[1] "super area"  "carpet area" "built area"
> unique(house_data$`Contact Person`)
[1] "contact owner"   "contact agent"   "contact builder"
```

Figure 3-24: Unique Values of Categorical Columns

The unique() function is utilized to retrieve unique elements from each categorical present by deleting duplicate elements and rows. Through the function, it assists in determining values for categorization purposes.

Related Extra Features:

Extra Feature 1: par()

3.3 Data Cleaning

Data cleaning involves the process of adjusting and removing inaccurate, corrupted, duplicated or noisy data within a dataset. If incorrect data is not treated, the outcomes in daa visualization is unreliable. There are a few basic steps taken to initiate data cleaning, which includes removing duplicates, fixing errors in structure, filtering outliers, handling missing data and validating the data (Tableau, n.d.).

1. **Change the data type of posting date from character to date**

Code:

```
#Change data type of posting date to date
house_data$`Posting Date` <- as.Date(house_data$`Posting Date`, format="%m-%d-%Y")
str(house_data)
```

Figure 3-25: Changing Data Types

Output:

```
> str(house_data)
'data.frame':   4746 obs. of  12 variables:
 $ Posting Date      : Date, format: "2022-05-18" "2022-05-13" "2022-05-16" "2022-07-04" ...
 $ BHK In Unit       : int  2 2 2 2 2 2 2 1 2 2 ...
 $ Rent Price        : int  10000 20000 17000 10000 7500 7000 10000 5000 26000 10000 ...
 $ Unit Size         : int  1100 800 1000 800 850 600 700 250 800 1000 ...
 $ Floor Number      : chr  "Ground out of 2" "1 out of 3" "1 out of 3" "1 out of 2" ...
 $ Type of Area      : chr  "Super Area" "Super Area" "Super Area" "Super Area" ...
 $ Locality          : chr  "Bandel" "Phool Bagan, Kankurgachi" "Salt Lake City Sector 2" "Dumdum Park" ...
 $ City Located      : chr  "Kolkata" "Kolkata" "Kolkata" "Kolkata" ...
 $ Furnishings       : chr  "Unfurnished" "Semi-Furnished" "Semi-Furnished" "Unfurnished" ...
 $ Preferred Tenants : chr  "Bachelors/Family" "Bachelors/Family" "Bachelors/Family" "Bachelors/Family" ...
 $ Number of Bathroom: int  2 1 1 1 1 2 2 1 2 2 ...
 $ Contact Person    : chr  "Contact Owner" "Contact Owner" "Contact Owner" "Contact Owner" ...
```

Figure 3-26: Changed Posting Date Data Type

To ensure accurate representation of data and improves the integrity of data, the data type of the posting date is converted to Date through the as.Date() function with the format of "MM-DD-YYYY". The structure of objects is called once again to ensure that the data type is changed to the desired outcome.

2. Inspect the number of missing values

Code:

```
#Number of missing values
colSums(is.na(house_data))
```

Figure 3-27: Inspect the Number of Missing Values

Output:

```
> #Number of missing values
> colSums(is.na(house_data))
      Posting Date        BHK In Unit         Rent Price
                 0                  0                  0
         Unit Size       Floor Number       Type of Area
                 0                  0                  0
          Locality       City Located        Furnishings
                 0                  0                  0
 Preferred Tenants Number of Bathroom     Contact Person
                 0                  0                  0
```

Figure 3-28: Number of Missing Values of Each Column

Missing values tend to be an obstacle to drawing accurate inferences about the data as it is deemed as a method to improperly handle data. The colSums() function is used along with the is.na() function to form row and column sums for the null values present in the house data. According to the results, there are no null values within the dataset.

3. Converting values to lower case for standardization

Code:

```
#Standardise the format of locality
house_data$Locality <- tolower(house_data$Locality)

#Maintain the consistency of data
house_data$`Floor Number` <- tolower(house_data$`Floor Number`)
house_data$`Type of Area`<- tolower(house_data$`Type of Area`)
house_data$`City Located` <- tolower(house_data$`City Located`)
house_data$Furnishings <- tolower(house_data$Furnishings)
house_data$`Preferred Tenants`<- tolower(house_data$`Preferred Tenants`)
house_data$`Contact Person`<- tolower(house_data$`Contact Person`)
```

Figure 3-29: Standardize Letter Casing

Output:

	Posting Date	BHK In Unit	Rent Price	Unit Size	Floor Number	Type of Area	Locality	City Located	Furnishings	Preferred Tenants	Number of Bathroom	Contact Person
1	2022-05-18	2	10000	1100	ground out of 2	super area	bandel	kolkata	unfurnished	bachelors/family	2	contact owner
2	2022-05-13	2	20000	800	1 out of 3	super area	phool bagan, kankurgachi	kolkata	semi-furnished	bachelors/family	1	contact owner
3	2022-05-16	2	17000	1000	1 out of 3	super area	salt lake city sector 2	kolkata	semi-furnished	bachelors/family	1	contact owner
4	2022-07-04	2	10000	800	1 out of 2	super area	dumdum park	kolkata	unfurnished	bachelors/family	1	contact owner
5	2022-05-09	2	7500	850	1 out of 2	carpet area	south dum dum	kolkata	unfurnished	bachelors	1	contact owner
6	2022-04-29	2	7000	600	ground out of 1	super area	thakurpukur	kolkata	unfurnished	bachelors/family	2	contact owner
7	2022-06-21	2	10000	700	ground out of 4	super area	malancha	kolkata	unfurnished	bachelors	2	contact agent
8	2022-06-21	1	5000	250	1 out of 2	super area	malancha	kolkata	unfurnished	bachelors	1	contact agent
9	2022-06-07	2	26000	800	1 out of 2	carpet area	palm avenue kolkata, ballygunge	kolkata	unfurnished	bachelors	2	contact agent
10	2022-06-20	2	10000	1000	1 out of 3	carpet area	natunhat	kolkata	semi-furnished	bachelors/family	2	contact owner
11	2022-05-23	3	25000	1200	1 out of 4	carpet area	action area 1, rajarhat newtown	kolkata	semi-furnished	bachelors/family	2	contact agent
12	2022-06-07	1	5000	400	1 out of 1	carpet area	keshtopur	kolkata	unfurnished	bachelors/family	1	contact agent
13	2022-05-14	1	6500	250	1 out of 4	carpet area	tarulia, keshtopur	kolkata	furnished	bachelors	1	contact owner
14	2022-05-09	1	5500	375	1 out of 2	carpet area	dum dum metro	kolkata	unfurnished	bachelors/family	1	contact agent
15	2022-05-05	3	8500	900	ground out of 2	carpet area	paschim barisha	kolkata	unfurnished	bachelors	2	contact owner
16	2022-06-01	3	40000	1286	1 out of 1	carpet area	new town action area 1	kolkata	furnished	bachelors/family	2	contact owner
17	2022-05-17	2	6000	600	1 out of 2	super area	barasat	kolkata	semi-furnished	bachelors/family	1	contact owner
18	2022-06-20	2	10000	800	ground out of 2	super area	behala	kolkata	unfurnished	bachelors/family	1	contact owner
19	2022-06-09	2	11000	2000	ground out of 3	carpet area	behala chowrasta	kolkata	unfurnished	bachelors/family	1	contact owner
20	2022-06-09	2	6000	660	1 out of 2	super area	behala	kolkata	unfurnished	bachelors/family	1	contact owner
21	2022-07-02	2	7900	650	1 out of 2	carpet area	santoshpur	kolkata	unfurnished	family	1	contact owner
22	2022-06-14	2	9000	400	2 out of 3	carpet area	garia station, garia	kolkata	unfurnished	bachelors	2	contact owner
23	2022-06-15	1	4000	300	ground out of 4	carpet area	garia station, garia	kolkata	unfurnished	bachelors/family	1	contact owner
24	2022-06-15	3	6500	1600	ground out of 2	super area	joka	kolkata	unfurnished	bachelors/family	1	contact owner
25	2022-05-28	1	8000	400	1 out of 2	super area	sreebhumi	kolkata	semi-furnished	bachelors	1	contact agent

Figure 3-30: Converted Letter Casings

The capitalized letter casing is converted to lower casing to ensure the consistency of data and to reduce noise in the data. The categorized columns such as Locality, Floor Number, Type of Area, City Located, Furnishings, Preferred Tenants and Contact Person is converted to lower case with the tolower() function. The output reflects the changes by displaying all values within the columns lower casing.

4. Trimming the leading and trailing whitespace

Code:

```
#Strip leading and trailing space
house_data$`Floor Number` <- trimws(house_data$`Floor Number`, which = c("both"))
house_data$`Type of Area` <- trimws(house_data$`Type of Area`, which = c("both"))
house_data$Locality <- trimws(house_data$Locality, which = c("both"))
house_data$`City Located` <- trimws(house_data$`City Located`, which = c("both"))
house_data$Furnishings <- trimws(house_data$Furnishings, which = c("both"))
house_data$`Preferred Tenants` <- trimws(house_data$`Preferred Tenants`, which = c("both"))
house_data$`Contact Person` <- trimws(house_data$`Contact Person`, which = c("both"))
```

Figure 3-31: Stripping Leading and Trailing Space

In order to further standardize the data, the leading and trailing whitespace is stripped off from variables that contains alphabetical values with the trimws() function, along with the parameter which = c(“both”) so that both spaces can be trimmed.

5. Identifying Outliers

Plot:

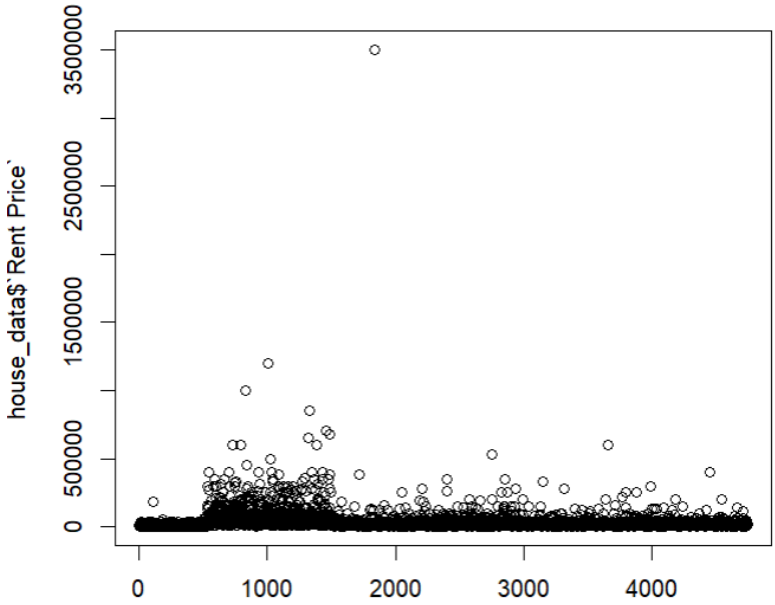

Figure 3-32: Scatterplot to Determine Outliers

Code:

```
lower_bound <- quantile(house_data$`Rent Price`, 0.1)
upper_bound <- quantile(house_data$`Rent Price`, 0.99)

outlier_ind <- which(house_data$`Rent Price` < lower_bound | house_data$`Rent Price` > upper_bound)
```

Figure 3-33: Detecting Outliers

Output:

```
> outlier_ind
  [1]    8   12   13   14   17   20   23   24   27   28   33   34   43
 [14]   45   47   53   57   58   61   65   67   69   74   77   78   79
 [27]   85   89   94   96  106  110  112  117  119  121  123  129  130
 [40]  131  132  134  137  147  149  152  153  160  163  169  171  176
 [53]  178  190  195  196  197  201  204  205  206  208  209  216  221
 [66]  222  223  227  229  232  236  238  239  241  250  251  253  257
 [79]  258  265  268  270  272  274  278  280  281  283  286  291  295
 [92]  299  307  319  321  322  324  331  332  333  351  352  355  357
[105]  362  365  370  376  379  380  384  387  389  397  399  401  402
[118]  405  408  412  413  415  417  418  419  420  421  427  428  429
[131]  430  432  434  437  439  440  441  443  444  445  450  451  453
[144]  468  470  472  477  482  485  489  495  501  504  507  508  509
[157]  511  513  520  544  590  632  667  703  712  727  743  758  793
[170]  828  840  909  928  937  945 1002 1024 1036 1038 1043 1087 1093
[183] 1225 1271 1276 1288 1320 1330 1345 1370 1385 1394 1412 1426 1432
[196] 1460 1477 1483 1485 1526 1528 1560 1573 1599 1603 1612 1628 1655
[209] 1662 1665 1705 1706 1719 1722 1734 1742 1745 1746 1750 1760 1769
[222] 1774 1777 1783 1802 1804 1838 1854 1866 1881 1891 1900 1901 1910
[235] 1914 1918 1923 1929 1948 1949 1969 1972 1982 2008 2019 2020 2023
[248] 2054 2058 2081 2083 2092 2096 2110 2112 2120 2138 2143 2171 2176
[261] 2183 2225 2229 2231 2239 2248 2255 2257 2269 2276 2287 2288 2294
[274] 2295 2325 2326 2337 2353 2354 2363 2364 2365 2392 2404 2448 2453
[287] 2459 2461 2464 2467 2470 2476 2549 2556 2561 2595 2628 2636 2641
[300] 2650 2654 2663 2681 2710 2721 2731 2732 2737 2751 2762 2770 2785
[313] 2812 2831 2858 2868 2874 2914 2936 2937 2940 2955 2956 2960 2970
[326] 2974 2989 3004 3014 3016 3024 3026 3030 3032 3065 3067 3068 3069
[339] 3085 3109 3121 3144 3149 3152 3162 3191 3193 3221 3245 3247 3272
[352] 3284 3288 3317 3335 3368 3377 3379 3395 3418 3430 3431 3433 3443
[365] 3460 3466 3470 3496 3515 3518 3565 3569 3590 3596 3613 3618 3657
[378] 3701 3712 3714 3734 3747 3752 3759 3765 3774 3777 3816 3863 3920
[391] 3925 3947 3960 3967 3988 3991 4013 4044 4062 4077 4085 4101 4109
[404] 4115 4119 4136 4142 4164 4178 4200 4202 4221 4244 4254 4271 4273
[417] 4295 4298 4315 4340 4341 4355 4416 4429 4432 4444 4455 4458 4461
[430] 4462 4474 4484 4486 4492 4493 4513 4514 4527 4548 4563 4583 4587
[443] 4606 4625 4634 4637 4638 4655 4659 4690 4706 4707 4710 4722 4734
```

Figure 3-34: Detected Outliers

The influence of outliers on the results of statistical tests is large, making it a necessity to search for outliers present within the dataset. The outliers might affect the normal distribution of the dataset, distort the standard deviation and mean of the columns involved. Therefore, the outliers in the Rent Price column are detected based on the percentiles. All of the observations that falls outside of the interval formed by the 1 and 99 percentiles (similar results with the IQR criterion) will be deemed as potential outliers. The quantile() function is used to compute the lower and upper percentiles. Then, the values outside the interval are extracted with the which() function. According to the outcome, 455 outliers are identified.

6. Replace Outliers

Code:

```
#Replace outliers
house_data$`Rent Price`[house_data$`Rent Price` %in% outlier_ind] <- median(house_data$`Rent Price`)
house_data[house_data$`Rent Price` == 3500000, ]$`Rent Price` <- median(house_data[house_data$`City Located` == "bangalore", ]$`Rent Price`)
```

Output:

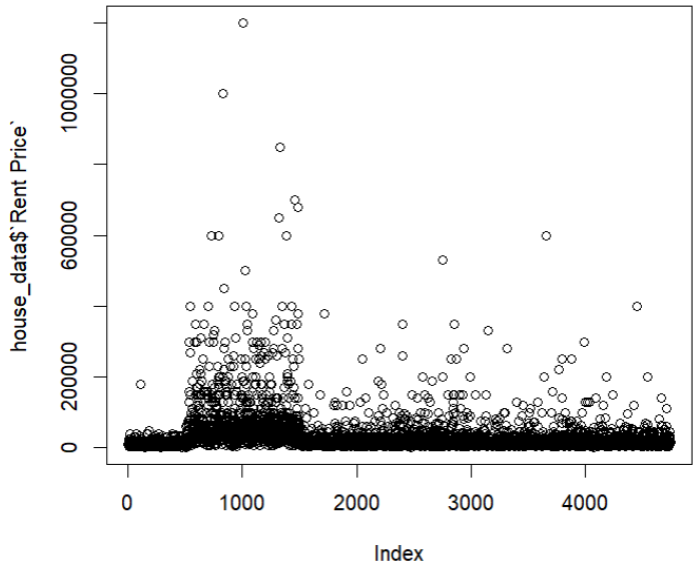

Figure 3-35: Scatterplot for Rent Price After Removing Outliers

The identified outliers are replaced with the median of the rent price as the value is less distorted compared to the mean value which is drastically affected by the outliers. The removal of outliers is not encouraged as it will increase the amount of insight loss as there are lesser data to be analyzed on. After replacing the outliers, the scatterplot shows a lesser range gap between the values compared to the plot at Figure 3-32.

Related Extra Features:

Extra Feature 2: duplicated()

Extra Feature 4: log()

3.4 Data Transformation

Data transformation is defined as the process of converting data from the initial format to a format that brings more meaning to the data (Pratt, n.d.). Data transformation helps in obtaining the maximum value of data, managing data in a more effective manner and improving the quality of data. The common forms of data transformation covers aggregation, construction of attributes, generalization, discretization, integration, manipulation, normalization and smoothing.

1. Transform price of each unit area

Code:

```
#Section 4: Data Transformation
#Price of each unit area
house_data$"Price Per Square Feet" <- round(house_data$`Rent Price`/ house_data$`Unit Size`, digits=2)
```

Figure 3-36: Calculate Price per Square Feet

Output:

Price Per Square Feet
9.09
25.00
17.00
12.50
8.82
11.67
14.29
20.00
32.50
10.00
20.83
12.50
26.00
14.67
9.44
31.10
10.00
12.50
5.50
9.09
12.15
22.50
13.33
4.06

Figure 3-37: Price per Square Feet Column

The Price per Square Feet column indicates the rent amount per square feet to dissect further insights on the rental pricing. The new column is transformed by dividing the rent price

with the unit size and rounding the values to two decimal places as prices are usually represented in two decimals.

2. Split Floor Number – Standardise Floor Number

Split Floor Number to Columns:

Extra Feature 8: str_split_fixed()

Code:

```
#Standardise the values of floor number
house_data["Floor"][house_data$Floor == "ground",] <- "0"
house_data["Floor"][house_data$Floor == "upper basement",] <- "-1"
house_data["Floor"][house_data$Floor == "lower basement",] <- "-2"
```

Figure 3-38: Standardising Values After Splitting

Output:

```
> unique(house_data$Floor)
 [1] "0"  "1"  "2"  "4"  "3"  "5"  "7"  "8"  "-1" "11" "-2" "6"  "14" "43" "13"
[16] "18" "17" "9"  "19" "60" "34" "12" "26" "25" "53" "16" "10" "39" "32" "47"
[31] "28" "20" "15" "65" "40" "37" "22" "21" "30" "35" "33" "44" "41" "46" "27"
[46] "45" "48" "50" "24" "23" "29" "49" "36" "76"
```

Figure 3-39: Standardised Floor Number

According to the dataset, the floors – ground, upper basement and lower basement are odd as it consists of alphabets rather than the majority which are integers. Hence, the data is standardized through the following logic:

Floor	**Floor Number After Conversion**
Ground	0
Upper Basement	-1
Lower Basement	-2

Table 3-2: Floor Number After Conversion

The process is done by specifying the condition of the data so that the desired value can be replaced according to the above conditions. After implementing the conversion, all floor numbers are standardized to integer representations in string form. This can ensure the consistency of data while visualization.

3. Split Floor Number – Replace Empty String

Inspection:

```
> unique(house_data$`Total Number of Floors`)
 [1] "2"  "3"  "1"  "4"  "5"  "14" "8"  "6"  "19" "10" "7"  "13" "78" "18" "12" "24" "31" "21" "23" "20" "9"
[22] "22" "58" "16" "66" "48" "40" "44" "42" "41" "60" "32" "30" "29" "89" "15" "11" "28" "17" "45" "35" "75"
[43] "38" "51" "43" "25" "27" "26" "76" "36" "37" "55" "68" "77" "50" "59" "62" "39" "52" "54" "33" "46" "85"
[64] "71" "81" "34" ""
```

Figure 3-40: Detection of An Empty String

Code:

```
#Replace empty values in total number of floors
house_data$`Total Number of Floors` <- as.numeric(house_data$`Total Number of Floors`)
total_floors <- house_data[!is.na(house_data$`Total Number of Floors`), ]
replace_total_floor <- median(total_floors$`Total Number of Floors`)
house_data["Total Number of Floors"][is.na(house_data$`Total Number of Floors`),] <- replace_total_floor
```

Figure 3-41: Replace Empty String

Output:

```
> house_data$`Total Number of Floors` <- as.numeric(house_data$`Total Number of Floors`)
> total_floors <- house_data[!is.na(house_data$`Total Number of Floors`), ]
> replace_total_floor <- median(total_floors$`Total Number of Floors`)
> house_data["Total Number of Floors"][is.na(house_data$`Total Number of Floors`),] <- replace_total_floor
> #Replace empty values in total number of floors
> unique(house_data$`Total Number of Floors`)
 [1]  2  3  1  4  5 14  8  6 19 10  7 13 78 18 12 24 31 21 23 20  9 22 58 16 66 48 40 44 42 41 60 32 30 29 89
[36] 15 11 28 17 45 35 75 38 51 43 25 27 26 76 36 37 55 68 77 50 59 62 39 52 54 33 46 85 71 81 34
>
```

Figure 3-42: Total Number of Floors After Replacement

The values of the total number of floors parsed is inspected with the unique() function to ensure that the data extracted is standardized. However, empty strings are detected from the data. To prevent errors and insight loss while visualization, the total number of floors are converted to numeric values. The value without the empty string is extracted and is stored under the total_floors variable so that no errors relating to missing values will be generated when the median is calculated. Then, total_floors is used to calculate and obtain the median of the total number of floors so that the empty values can be replaced. According to the outcome, the empty strings are all replaced.

Related Extra Features:

Extra Feature 3: gsub()

Extra Feature 5: ggcorrplot()

Extra Feature 8: str_split_fixed()

4.0 Question Analysis

4.1 Question 1: What is the preference of a bachelor in choosing a house?

4.1.1 Analysis 1-1: Determine the number of bachelors

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The number of bachelors present is explored

Code:

```
#Question 1: What is the preference of a bachelor in choosing a house?

#Number of Bachelors
number_of_bachelors = nrow(house_data[(house_data$`Preferred Tenants` == "bachelors")
                                      |(house_data$`Preferred Tenants` == "bachelors/family"),])
number_of_bachelors
```

Figure 4-1: Calculating the Number of Bachelors

Output:

```
> number_of_bachelors
[1] 4274
```

Figure 4-2: Total Number of Bachelors

Before initiating the analysis process, the number of bachelors is calculated to understand the approximate rows of data relating to bachelors. To calculate the number of bachelors, the number of rows that meets the criteria of having either bachelors or bachelors/family within the Preferred Tenants column is selected for calculation. According to the outcome, there is a total of 4274 bachelors within the dataset.

4.1.2 Analysis 1-2: Determine the preference of BHK by most bachelors

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The distribution of the number of bachelors for each BHK is explored
Data Visualization	The result of the distribution of BHK within bachelors are visualized in the form of bar chart

Source code:

```
# For bachelors
calculate_bhk <- function(bhk)
{
  nrow(house_data[(house_data$`BHK In Unit` == bhk) &
                    ((house_data$`Preferred Tenants` == "bachelors") |
                       (house_data$`Preferred Tenants` == "bachelors/family")),])
}

one_bhk <- calculate_bhk(1)
two_bhk <- calculate_bhk(2)
three_bhk <- calculate_bhk(3)
four_bhk <- calculate_bhk(4)
five_bhk <- calculate_bhk(5)
six_bhk <- calculate_bhk(6)

#Plot barplot to determine the most BHK preferred
bhk_number <- c(one_bhk,two_bhk,three_bhk,four_bhk,five_bhk,six_bhk)
bhk_label <- c("1","2","3","4","5","6")
bachelor_bhk <- barplot(bhk_number, main = "Distribution of BHK In Unit for Bachelors", xlab = "Number of BHK",
        ylab = "Number of Bachelors", names.arg = bhk_label, ylim = c(0,3000), col = c("darkturquoise", "deeppink", "darkorchid",
                                                                                       "deepskyblue", "cyan4", "blueviolet"))
text(bachelor_bhk, 0, bhk_number, cex = 1, pos = 3)
```

Figure 4-3: Calculating and Visualizing the Number of Bachelor per BHK

The calculate_bhk() function is defined to calculate the number of rows present according to the conditions of the preferred tenants being either bachelors or bachelors/family. The bhk variable is being passed as an argument so that the function can accommodate various bhk for calculation. The function is then called to the variables – one_bhk, two_bhk, three_bhk, four_bhk, five_bhk and six_bhk with the values 1,2,3,4,5 and 6 as the argument for the respective functions.

After calculating the number of bachelors for each BHK, a vector is defined as bhk_number to store the values of each BHK, along with the labels according to the BHK number. A barplot is plotted by passing the bhk_number vector as the x-axis, with “Distribution of BHK In Unit for Bachelors” as the title, “Number of BHK” as the x-axis label, “Number of Bachelors” as the y-axis label, the labels of the x-axis variables as bhk_label and a vector of colors is defined to better distinguish the difference between the BHK values. The text()

function is called to visualize the values of each BHK on the barplot to further execute comparisons between the BHK values.

Plot:

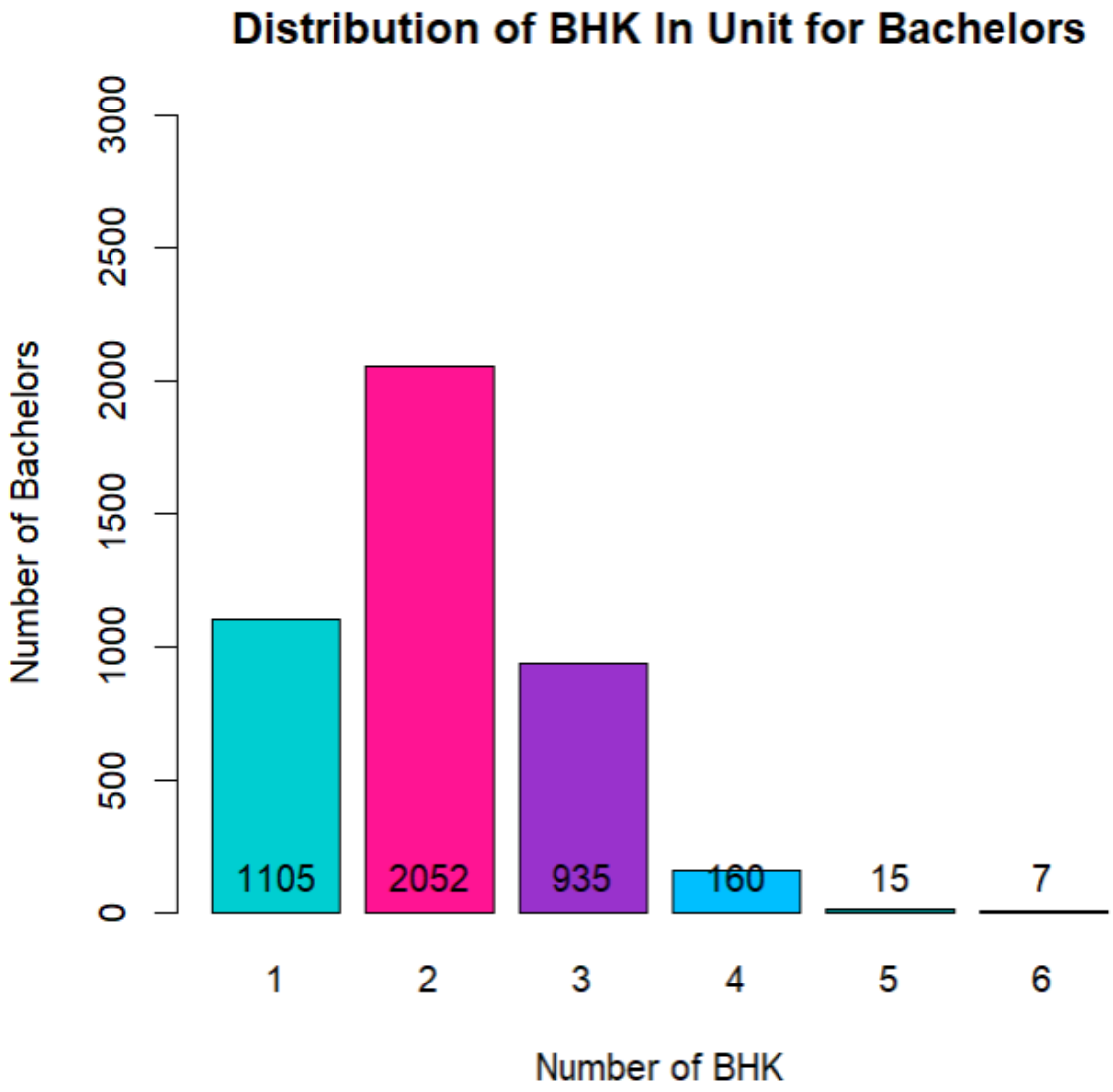

Figure 4-4: Distribution of BHK In Unit for Bachelors

Based on the bar plot, 2052 bachelors prefer to have houses with 2 BHK, which is also deemed to be the most preferred BHK. Houses with 6 BHK tend to be the least preferred type of house among bachelors as it only consists of 7 of them. The difference between the number of bachelors for the most and least preferred BHK is 2045. Bachelors prefer a 2 BHK house might be due to the suitability of the number of bedrooms, and they can also share the house with their friends.

4.1.3 Analysis 1-3: Determine the preference of the number of bathrooms of the most preferred BHK

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The distribution of the number of bachelors for each bathroom for 2 BHK is explored
Data Visualization	The result of the distribution of the number of bathrooms for 2 BHK within bachelors are visualized in the form of bar chart

Source Code:

```
#Analysis 1-2: How many bathrooms do usually the most preferred BHK has?
#Distribution of the number of bathrooms
house_data %>% count(`Number of Bathroom`)

calculate_bathroom <- function(bathroom)
{
  nrow(house_data[(house_data$`Number of Bathroom` == bathroom) &
                    (house_data$`BHK In Unit` == 2),])
}

one_bathroom <- calculate_bathroom(1)
two_bathroom <- calculate_bathroom(2)
three_bathroom <- calculate_bathroom(3)
four_bathroom <- calculate_bathroom(4)
five_bathroom <- calculate_bathroom(5)
six_bathroom <- calculate_bathroom(6)
seven_bathroom <- calculate_bathroom(7)
ten_bathroom <- calculate_bathroom(10)

#Plot barplot for bathrooms of 2 BHK units
bathroom_number <- c(one_bathroom, two_bathroom, three_bathroom, four_bathroom,
                     five_bathroom, six_bathroom, seven_bathroom, ten_bathroom)
bathroom_label <- c("1", "2", "3", "4", "5", "6", "7", "10")
bathroom <- barplot(bathroom_number, main = "Distribution of The Number of Bathrooms for 2 BHK",
        xlab = "Number of Bathroom", ylab = "Number of Bachelors", ylim = c(0, 2000), names.arg = bathroom_label,
        col = c("darkturquoise", "deeppink", "darkorchid","deepskyblue", "cyan4", "blueviolet", "blue", "aquamarine"))
text(bathroom, 0, bathroom_number, cex = 1, pos = 3)
```

Figure 4-5: Calculating and Visualizing the Number of Bachelor per 2 BHK Bathrooms

The calculate_bathroom() function is defined to calculate the number of rows present according to the conditions of the preferred tenants being either bachelors or bachelors/family. The bathroom variable is being passed as an argument so that the function can accommodate various number of bathrooms for calculation. The function is then called to the variables – one_bathroom, two_bathroom, three_bathroom, four_bathroom, five_bathroom and six_bathroom with the values 1,2,3,4,5,6,7 and 10 as the argument for the respective functions.

After calculating the number of bachelors for each number of bathrooms, a vector is defined as bathroom_number to store the values of each number of bathrooms, along with the labels according to the number of bathrooms. A barplot is plotted by passing the

bathroom_number vector as the x-axis, with “Distribution of The Number of Bathrooms for 2 BHK” as the title, “Number of Bathroom” as the x-axis label, “Number of Bachelors” as the y-axis label, the labels of the x-axis variables as bathroom_label and a vector of colors is defined to better distinguish the difference between the values for the number of bathrooms. The text() function is called to visualize the values of each number of bathrooms on the barplot to further execute comparisons between the values.

Plot:

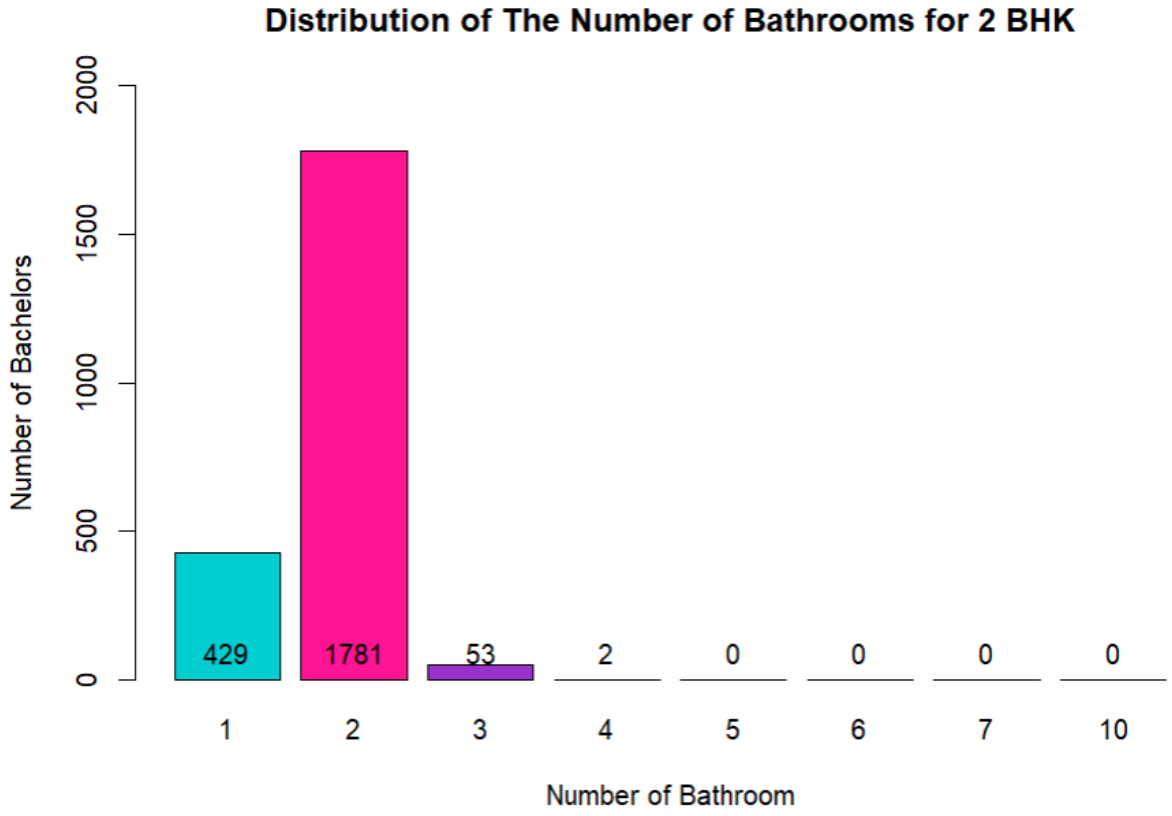

Figure 4-6: Distribution of the Number of Bathrooms for 2 BHK

The bar plot states that 1781 bachelors prefer to have 2 bathrooms in a 2 BHK unit, followed by 1 bathroom with 429 bachelors and 3 bathrooms with 53 bachelors. However, bachelors do not prefer to have more than 3 bathrooms in their unit and prefer to stick to lesser number of bathrooms. For 2 BHK houses, 2 bathrooms are usually ideal as an individual of each bedroom obtains more privacy rather than having to share some bathroom appliances together.

4.1.4 Analysis 1-4: Determine the average rent preferred by bachelors

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The rent price of each bachelor is extracted
Data Manipulation	The mean of the rent price of bachelors are calculated

Source Code:

```
#Analysis 1-3: What are the average rent price preferred by bachelors?

bachelor_rent_price <- mean(house_data[(house_data$`Preferred Tenants`== "bachelors"|
                                          house_data$`Preferred Tenants`== "bachelors/family"),
                                        'Rent Price'])
bachelor_rent_price
```

Figure 4-7: Calculate the Average Rent Price for Bachelors

The rent price of rows that fulfills the condition of having either bachelors or bachelors/family as the preferred tenants are selected so that the mean can be calculated, which is then presented as the average rent price.

Output:

```
> bachelor_rent_price
[1] 33333.95
```

Figure 4-8: Average Rent Price for Bachelors

According to the output, the average rent price that bachelors prefer are 33333.95 INR.

4.1.5 Analysis 1-5: Determine the average unit size preferred by bachelors

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The unit size of each bachelor is extracted
Data Manipulation	The mean of the unit size of bachelors is calculated

Source Code:

```
#Analysis 1-4: What are the average unit size preferred by bachelors?

bachelor_unit_size <- mean(house_data[(house_data$`Preferred Tenants` == "bachelors" |
                                       house_data$`Preferred Tenants` == "bachelors/family"),
                                      'Unit Size'])

bachelor_unit_sizes
```

Figure 4-9: Calculate the Average Unit Size for Bachelors

The unit size of rows that fulfills the condition of having either bachelors or bachelors/family as the preferred tenants are selected so that the mean can be calculated, which is then presented as the average unit size.

```
#Range of unit size
ggplot(house_data, aes(y=`Unit Size`, x=`Preferred Tenants`, color=`Preferred Tenants`)) +
  geom_boxplot() +
  labs(title="Range of Unit Size According to Tenants")
```

Figure 4-10: Plotting Boxplot for The Range of Unit Size

The boxplot is plotted with ggplot() where the unit size is represented as the y-axis and preferred tenants is represented as the x-axis from the house_data dataset variable. The color of the boxplot is represented according to the preferred tenants.

Output:

```
> bachelor_unit_size
[1] 946.7466
```

Figure 4-11: Average Unit Size for Bachelors

Based on the output, the average unit price that bachelors prefer are 946.7466 square feet. They do not prefer a larger size house as they usually live alone or with 1 or 2 companions.

Plot:

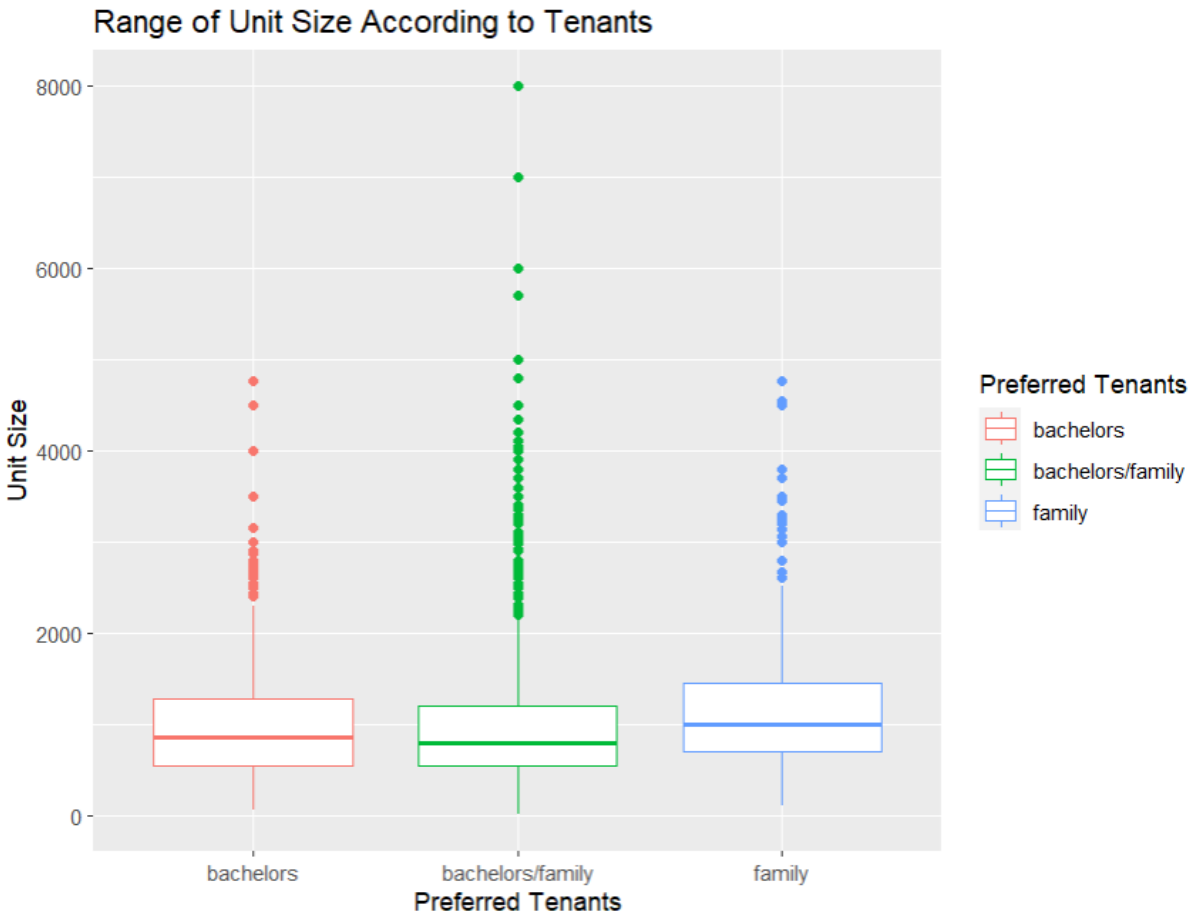

Figure 4-12: Range of Unit Size According to Tenants

The box plot for family is the highest compared to bachelors and bachelors/family with the highest median present. However, the range of the unit size holds only a slight difference between each preferred tenant. The box plot for all tenants is relatively short, hence it suggests that the range of unit size is small.

4.1.6 Analysis 1-6: Determine the preference of bachelors on lower or upper floors

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The overall statistical summary of the floor column is explored
Data Manipulation	A new column – Floor Range is derived with new values introduced according to the details obtained from exploration
Data Visualization	The distribution of floor ranges is visualized in the form of pie chart

Source Code:

```
floor_integer <- as.integer(house_data$Floor)
summary(floor_integer)
individual_floor <- house_data %>% distinct(floor_integer, keep_all = TRUE)
#Median = 24.5
summary(individual_floor)

#Groups the floors into lower floors and upper floors range
#Median is rounded up to 25 as floors cannot be counted in decimals
house_data <- house_data %>%
  mutate("Floor Range"= if_else(house_data$Floor <= 25, 'lower floors',
                                'upper floors'))

#Pie chart for floor range
lower_floor = nrow(house_data[(house_data$`Floor Range` == "lower floors") &
                              ((house_data$`Preferred Tenants` == "bachelors")
                               | (house_data$`Preferred Tenants` == "bachelors/family")),])
upper_floor = nrow(house_data[house_data$`Floor Range` == "upper floors" &
                              ((house_data$`Preferred Tenants` == "bachelors")
                               | (house_data$`Preferred Tenants` == "bachelors/family")),])

floor = c(lower_floor, upper_floor)
label = c(lower_floor, upper_floor)

pie(floor, label, radius = 1, main="Distribution of Floor Range", col = c("cyan", "lightblue"))
legend("topright", c("Lower Floor", "Upper Floor"), cex = 0.7, fill = c("cyan", "lightblue"))
```

Figure 4-13: Calculate Floor Data

The floor column is first converted to integer so that the statistical summary of the column can be generated. Through the summary provided, it is decided that the distinct values of the floor column should be extracted so that no duplicated values are present to interrupt the results for statistical calculations. According to the summary, the median of the floor is 24.5. However, the value should be rounded up to 25 as floor is deemed as a discrete value. With the help of the mutate() function, the new column – Floor Range is derived through categorizing the floors to lower floors and upper floors. If the floor number is below the median, it will be

categorized as the lower floors whereas floor numbers that are higher than the median are upper floors. Before plotting the pie chart, the number of rows for lower floors and upper floors are calculated respectively. According to the values obtained, a pie chart is plotted according to the floor range, the parts of the pie chart is labelled with the help of legends.

Output After Data Manipulation:

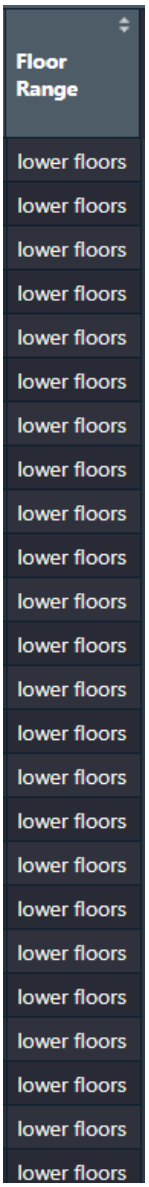

Figure 4-14: Floor Range Column

The new column consists of 2 categories – lower floors and upper floors.

Pie Chart:

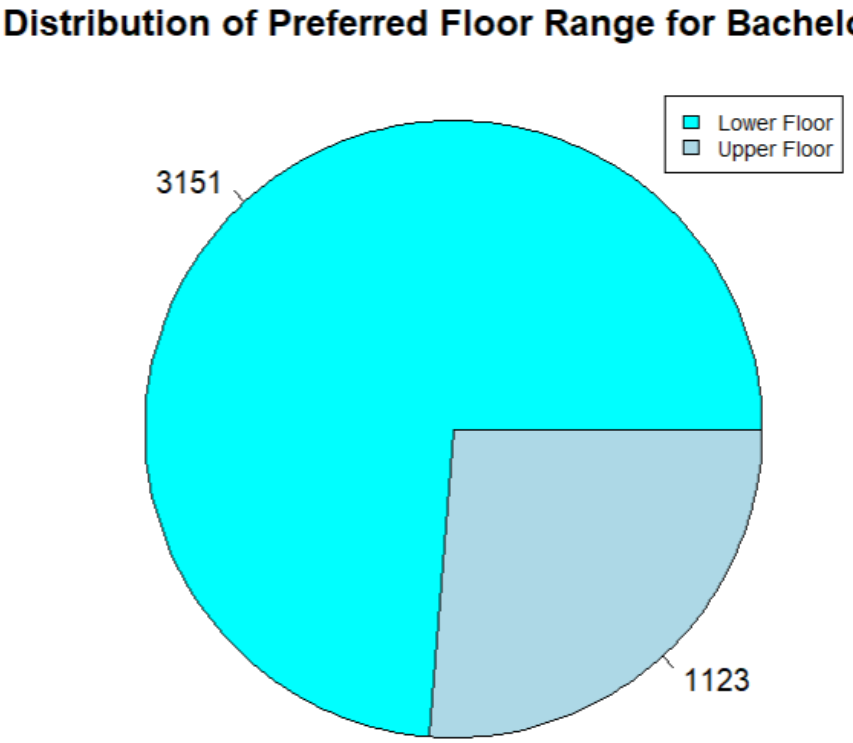

Figure 4-15: Distribution of Preferred Floor Range for Bachelors

The pie chart states that the distribution of the lower floor is higher than the upper floor, with the difference of 2028 bachelors. 3151 bachelors prefer to stay at lower floors whereas 1123 bachelors prefer to reside at upper floors. This could be due to the high convenience of accessing on-ground amenities and having a swifter evacuation route for lower floor units.

4.1.7 Analysis 1-7: Determine the preference of cities for bachelors

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The number of bachelors that resides at each city is explored
Data Visualization	A pie chart is plotted to present the distribution of the number of bachelors throughout the cities

Source Code:

```
#Analysis 1-6: Which cities do bachelors prefer to stay at?

calculate_city <- function(city)
{
  nrow(house_data[(house_data$`City Located` == city) &
                    ((house_data$`Preferred Tenants` == "bachelors") |
                       (house_data$`Preferred Tenants` == "bachelors/family")),])
}

bachelor_kolkata <- calculate_city("kolkata")
bachelor_mumbai <- calculate_city("mumbai")
bachelor_bangalore <- calculate_city("bangalore")
bachelor_delhi <- calculate_city("delhi")
bachelor_chennai <- calculate_city("chennai")
bachelor_hyderabad <- calculate_city("hyderabad")

bachelor_cities = c(bachelor_kolkata, bachelor_mumbai, bachelor_bangalore,
                    bachelor_delhi, bachelor_chennai, bachelor_hyderabad)
city_label = c("Kolkata", "Mumbai", "Bangalore", "Delhi", "Chennai", "Hyderabad")
city_number = c(bachelor_kolkata, bachelor_mumbai, bachelor_bangalore,
                bachelor_delhi, bachelor_chennai, bachelor_hyderabad)
color = c("red", "orange", "yellow", "green", "blue", "purple")

pie(bachelor_cities, city_number, radius = 1, main = "Distribution of Cities for Bachelors", col=color)
legend("topleft", city_label, cex = 0.7, fill=color)
```

Figure 4-16: Calculating and Visualizing the Number of Bachelor Residing in Each City

The calculate_city() function is defined to calculate the number of rows according to the criteria of selecting the city located based on the argument and bachelors or bachelors/family as preferred tenants. The function is then assigned by the arguments – kolkata, mumbai, bangalore, delhi, chennai and hyderabad as strings respectively to calculate the number of bachelors present within each city. The number of bachelors for each city are then stored in a vector with the name of bachelor cities, followed by initializing three more vectors to store the labels for the cities and the colors assigned. A pie chart is plotted with the pie() function along with the help of legend() to describe the cities. The pie chart is plotted according to the bachelor_cities vector to determine the distribution of bachelors renting preference within each city.

Pie Chart:

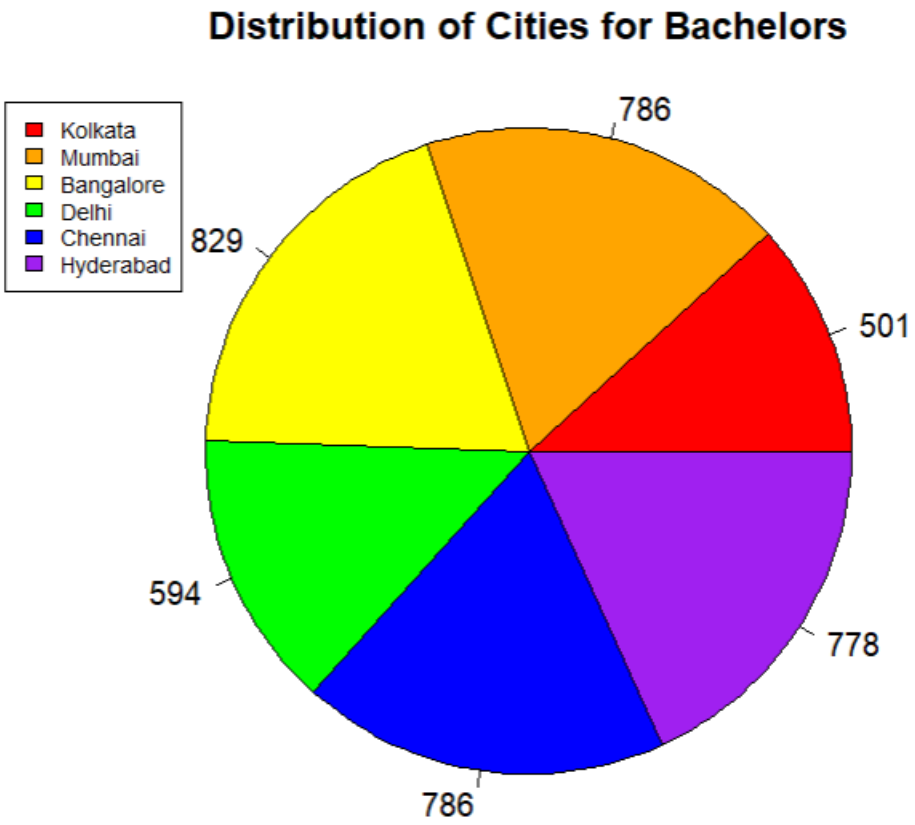

Figure 4-17: Distribution of Cities for Bachelors

Based on the pie chart, Bangalore has the highest number of distributions (829) in terms of preferences from bachelors, whereas Kolkata holds the least preference (501), where the difference between both cities is 328. The is an equal number of bachelors that prefers to reside at Mumbai and Chennai. Bachelors might prefer Bangalore due to the cheap cost of living, a balanced climate and good job opportunities (Houseey Tips, 2020).

4.1.8 Analysis 1-8: Determine the preference of the average total number of floors and the category of floors

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The number of bachelors that prefers different types of building height is explored
Data Manipulation	The mean and median of the total number of floors is calculated and a new column is derived according to the values
Data Visualization	A pie chart relating to the distribution of building height for bachelors are plotted

Source Code:

```
#Analysis 1-7: What are the average total number of floors bachelors prefer their rental building to be?
#              Do they prefer taller or shorter buildings?

floor_total <- as.integer(house_data$`Total Number of Floors`)
floor_mean <- ceiling(mean(house_data[(house_data$`Preferred Tenants`== "bachelors"|
                                         house_data$`Preferred Tenants`== "bachelors/family"),
                                       'Total Number of Floors']))
floor_mean

#The average total number of floors bachelors prefer to rent is 7 floors

floor_median <- ceiling(median(house_data[(house_data$`Preferred Tenants`== "bachelors"|
                                           house_data$`Preferred Tenants`== "bachelors/family"),
                                         'Total Number of Floors']))
house_data <- house_data %>%
  mutate("Building Category"= if_else(house_data$`Total Number of Floors`<= floor_median, 'short building',
                                      'tall building'))

#Pie chart for total floor range
short_building = nrow(house_data[(house_data$`Building Category` == "short building") &
                                   (house_data$`Preferred Tenants` == "bachelors" |
                                      house_data$`Preferred Tenants` == "bachelors/family"),])
tall_building = nrow(house_data[(house_data$`Building Category` == "tall building") &
                                   (house_data$`Preferred Tenants` == "bachelors" |
                                      house_data$`Preferred Tenants` == "bachelors/family"),])

building = c(short_building, tall_building)
building_label = c("Short Building", "Tall Building")
building_numbers = c(short_building, tall_building)

pie(building, building_numbers, radius = 1, main = "Distribution of Building Height For Bachelors",
    col = c("green", "lightgreen"))
legend("topright", building_label, cex = 0.7, fill = c("green", "lightgreen"))
```

Figure 4-18: Calculate Total Number of Floors Data

The mean of the total number of floors present for bachelors or bachelors/family is calculated and is rounded up as floors cannot exist in decimals. A similar approach is done for the median to determine the splitting point of short and tall buildings. According to the median calculated, a new column – Building Category is created by placing buildings with the total number of floors below the median as short building whereas tall building for building levels

above the median. The number of rows of each building category for bachelors are calculated and a pie chart is plotted to illustrate the distribution.

Output After Data Manipulation:

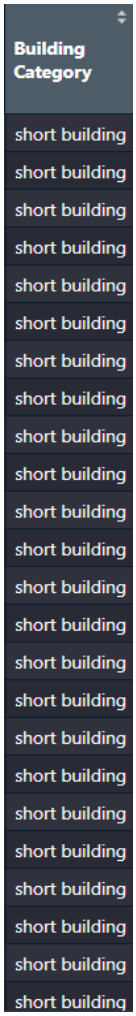

Figure 4-19: Building Category Column

The building category column consists of 2 categories – short building and tall building.

Mean Output:

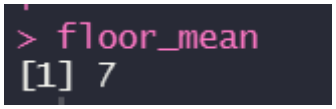

Figure 4-20: Mean Total Number of Floors for Bachelors

The mean of the total number of floors preferred by bachelors is 7 floors, which is higher than the median value of 4 floors.

Pie Chart:

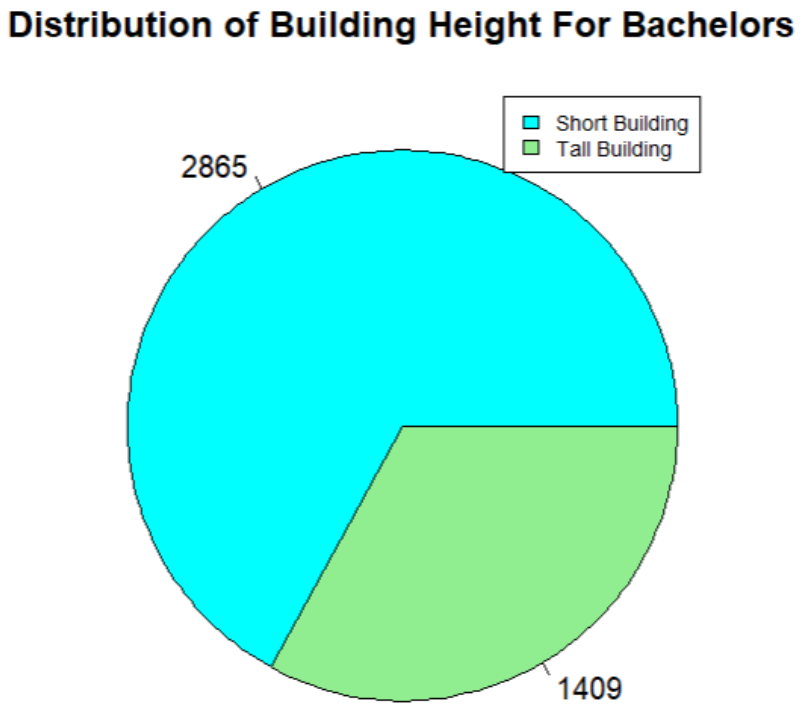

Figure 4-21: Distribution of Building Height for Bachelors

Most bachelors prefer to stay at short buildings rather than tall buildings, with the total difference of 1456 bachelors between the two types of building categories. Lower buildings provide bachelors with greater peace and privacy and is more convenient to access as it is located closer to the street.

4.1.9 Analysis 1-9: Determine the preference of the area type for bachelors

Analysis techniques used:

Analysis Technique	Reason
Data Exploration	The number of rows of bachelors for each area type is explored
Data Visualization	A bar chart is plotted according to the number of bachelors for each area type

Source Code:

```
#Analysis 1-8: Which type of area do most bachelors rent with?

bachelor_area <- function(area)
{
  nrow(house_data[(house_data$`Type of Area` == area) &
                    (house_data$`Preferred Tenants` == "bachelors" |
                       house_data$`Preferred Tenants` == "bachelors/family"),])
}

super_area = bachelor_area("super")
carpet_area = bachelor_area("carpet")
built_area = bachelor_area("built")

#Plot bar chart for type of area
area_type <- c(super_area,carpet_area,built_area)
area_label <- c("Super Area", "Carpet Area", "Built Area")

area <- barplot(area_type, main = "Distribution of The Area Types for Bachelors",
        xlab = "Type of Area", ylab = "Number of Bachelors", names.arg = area_label,
        col = c("darkturquoise", "deeppink", "darkorchid"), ylim = c(0,3000))
text(area, 0, area_type, cex = 1, pos = 3)
```

Figure 4-22: Calculate and Visualize the Distribution of Area Types for Bachelors

The bachelor_area() function is defined to calculate the number of rows present for bachelors or bachelors/family according to the type of area selected. The function is called for the super_area, carpet_area and built_area variables along with passing the arguments – super, carpet and built respectively in the form of strings. Then, a bar chart is plotted according to the outcome of the variables, which is stored in a vector.

Plot:

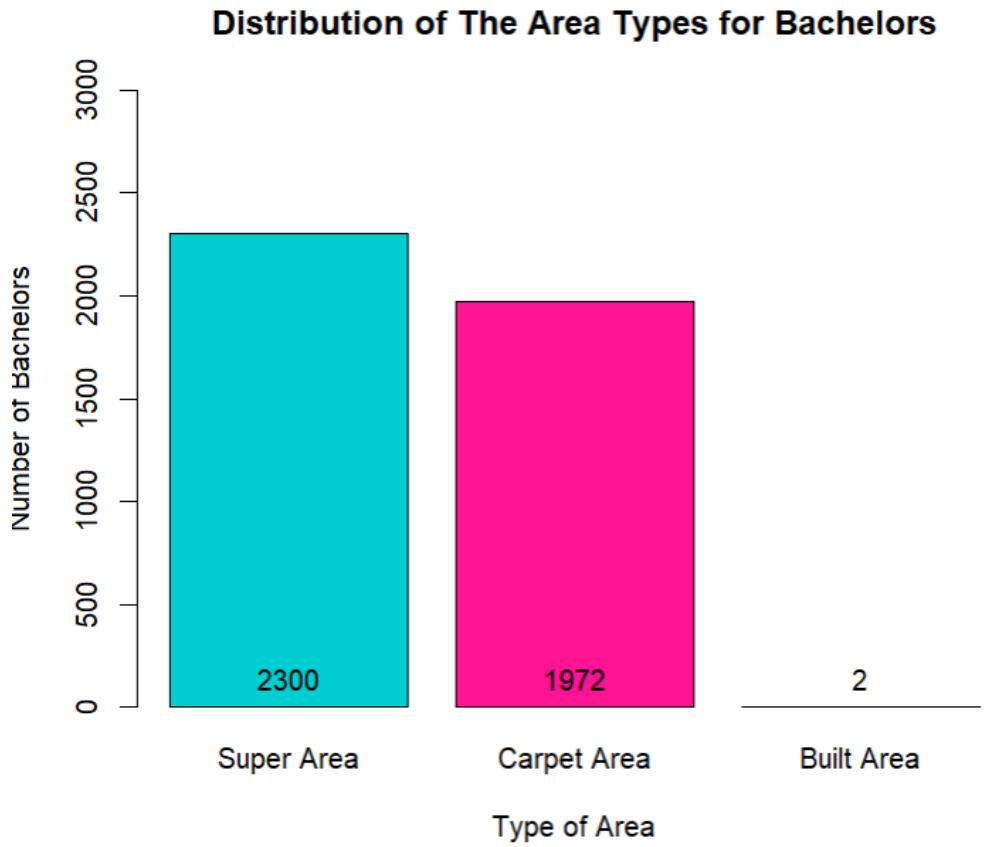

Figure 4-23: Distribution of Area Types for Bachelors

According to the bar plot, bachelors tend to prefer super area the most, followed by carpet area and build area. It can also be deduced that the popularity of built areas is significantly lower than the other area types, this might be because that the area includes walls that are shared with other units factored at 50 percent, which might not be that ideal in terms of pricing. Bachelors only consider the usable area of the house, which excludes the thickness of the walls.

4.1.10 Analysis 1-10: Determine the preference of the furnishing type for bachelors

Analysis techniques used:

Analysis Technique	Reason
Data Exploration	The number of rows of bachelors for each furnishing type is explored
Data Visualization	A bar chart is plotted according to the number of bachelors for each furnishing type

Source Code:

```
#Analysis 1-9: Which type of furnishings do most bachelors prefer?

bachelor_furnishing <- function(furnishing)
{
  nrow(house_data[(house_data$Furnishings == furnishing) &
                    (house_data$`Preferred Tenants`== "bachelors" |
                       house_data$`Preferred Tenants` == "bachelors/family"),])
}

unfurnished = bachelor_furnishing("unfurnished")
semi_furnished = bachelor_furnishing("semi-furnished")
furnished = bachelor_furnishing("furnished")

#Plot bar chart for furnishings
furnishing_type <- c(unfurnished, semi_furnished, furnished)
furnishing_label <- c("Unfurnished", "Semi-furnished", "Furnished")

furnishing <- barplot(furnishing_type, main = "Distribution of Furnishings for Bachelors",
                      xlab = "Type of Furnishing", ylab = "Number of Bachelors", names.arg = furnishing_label,
                      col = c("purple", "turquoise", "yellow"), ylim = c(0,3000))
text(furnishing, 0, furnishing_type, cex = 1, pos = 3)
```

Figure 4-24: Calculate and Visualize the Distribution of Furnishing Types for Bachelors

The bachelor_furnishing() function is defined to calculate the number of rows present for bachelors or bachelors/family according to the type of furnishing selected. The function is called for the unfurnished, semi_furnished and furnished variables along with passing the arguments – unfurnished, semi-furnished and furnished respectively in the form of strings. Then, a bar chart is plotted according to the outcome of the variables, which is stored in a vector.

Plot:

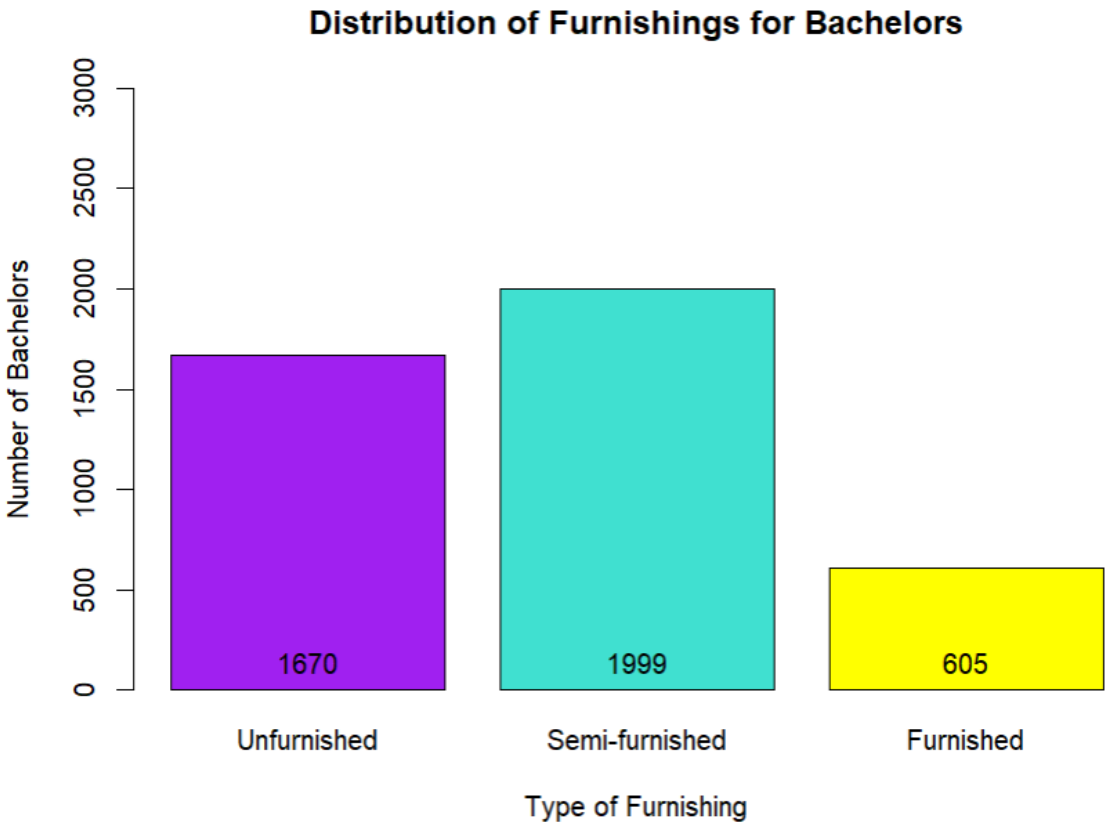

Figure 4-25: Distribution of Furnishings for Bachelors

The bar plot illustrates that 1999 bachelors, which is the majority, prefers to rent semi-furnished units over unfurnished and furnished units. It is suggested that bachelors might prefer such units due to the higher level of customization with basic appliances which they can adapt easily. Furnished units usually come with a higher price and most bachelors are unwilling to pay high rent.

4.1.11 Analysis 1-11: Determine the preference of bachelors in terms of contact person

Analysis techniques used:

Analysis Technique	Reason
Data Exploration	The number of rows of bachelors for each contact person is explored
Data Visualization	A bar chart is plotted according to the number of bachelors for each contact person

Source Code:

```
#Analysis 1-10: Who do bachelors usually reach out to?

bachelor_contact <- function(person)
{
  nrow(house_data[(house_data$`Contact Person` == person)
                  & (house_data$`Preferred Tenants`== "bachelors" |
                       house_data$`Preferred Tenants` == "bachelors/family"),])
}

owner = bachelor_contact("owner")
agent = bachelor_contact("agent")
builder = bachelor_contact("builder")

#Plot bar chart for contact person
contact_person <- c(owner, agent, builder)
contact_label <- c("Owner", "Agent", "Builder")

contact <- barplot(contact_person, main = "Distribution of Contact Person for Bachelors",
                   xlab = "Person of Contact", ylab = "Number of Bachelors", names.arg = contact_label,
                   col = c("purple3", "cyan", "blue"), ylim = c(0,3000))
text(contact, 0, contact_person, cex = 1, pos = 3)
```

Figure 4-26: Calculate and Visualize the Distribution of Contact Person for Bachelors

The bachelor_contact() function is defined to calculate the number of rows present for bachelors or bachelors/family based on the contact person selected. The function is called for the owner, agent and builder variables along with passing the arguments – owner, agent and builder respectively in the form of strings. Then, a bar chart is plotted according to the outcome of the variables, which is stored in a vector.

Plot:

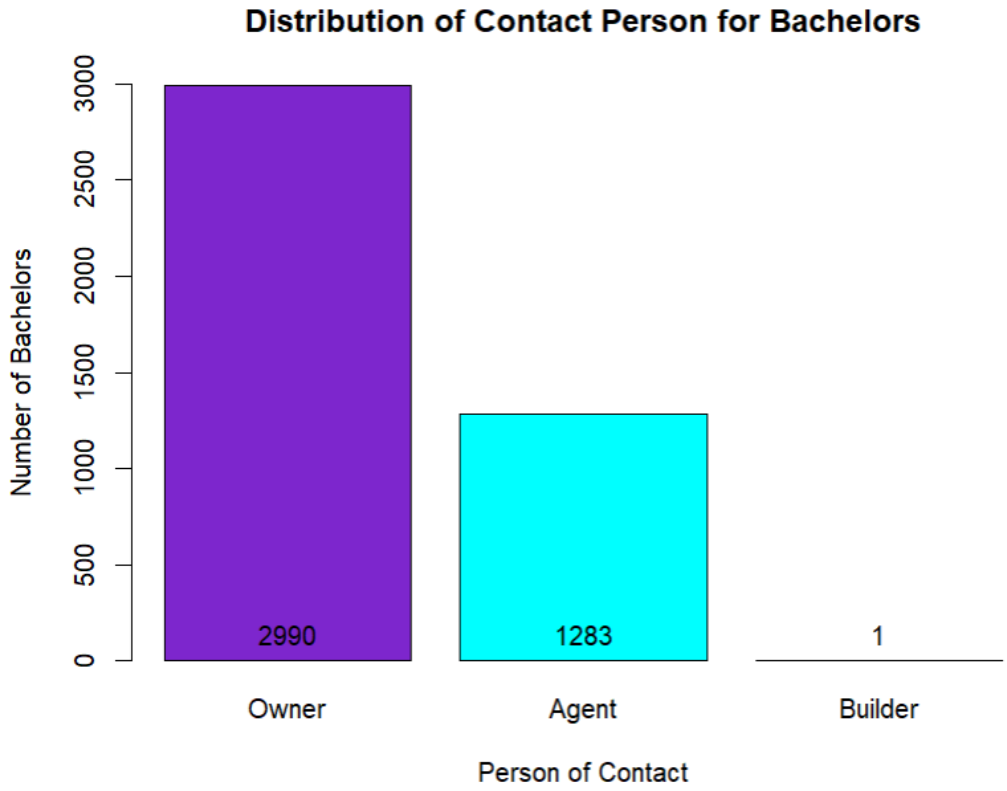

Figure 4-27: Distribution of Contact Person for Bachelors

The bar plot shows that most bachelors prefer to contact owners of the unit they wish to rent, followed by agents and builder. Contact with unnecessary middlemen during the rent inquiry process can be avoided if bachelors approach the owner directly and potentially obtain better deals from them, rather than contacting a builder which is deemed to be a less trustworthy way for enquiring the unit.

4.1.12 Conclusion

Through analyzing the preference of bachelors, it can be concluded that bachelors consider many factors in selecting their desired house for rental. The most popular preference of house is a unit with 2 BHK which consists of the following characteristics:

1. 2 bathrooms
2. Has an average rent price of 33333.95 INR
3. A unit size of 946.75 square feet
4. Located at the lower floors
5. Shorter buildings
6. Located in Bangalore
7. Super areas
8. Semi-furnished units
9. Having the owner as the main contact person

The characteristics stated has also determined that bachelors prefer units that are convenient to access, practical, budget, moderate sized, located in areas with a lower cost of living, has a high level of customization and has a contact person that is more convenient to enquire the unit and have negotiations.

On the other hand, bachelors put the least preferences on 7 BHK units with the following characteristics:

1. Has 4 or more bathrooms
2. Unit sizes more than 1200 square feet
3. Located at upper floors
4. Taller buildings
5. Located in Kolkata
6. Built areas
7. Furnished units
8. Having the builder as the main contact person

Bachelors do not prefer units that are deemed impractical for themselves to live in, the unit is considered as impractical when it has too many bathrooms, big area sizes, inconvenient to access, lower levels of customization in terms of their furniture which also provides the potential to pay higher rental fees. Kolkata is not preferred by bachelors as it has lesser job opportunities due to the backward development caused by political instability.

4.2 Question 2: What is the pricing of the rent affected by? Are the houses affordable?

4.2.1 Analysis 2-1: Relationship between unit size and rent

Analysis techniques used:

Analysis Technique	Reason
Data Visualization	A scatterplot is plotted to determine the relationship between the unit size and rent price with the help of a linear regression lines

Source Code:

```
#Question 2: What is the pricing of the rent affected by?
#Analysis 2-1: Relationship between unit size and rent.

ggplot(house_data, aes(x=`Unit Size`, y=`Rent Price`)) +
  geom_point(shape = 15, aes(colour = `Rent Price`)) + geom_smooth(method = lm) +
  scale_colour_gradient(low = "green", high = "yellow") +
  labs(title="Relationship Between Unit Size and Rent Price")
```

Figure 4-28: Determining the Relationship Between Unit Size and Rent

To determine the relationship between the unit size and rent, a scatterplot is plotted with the Unit Size as the x-axis and Rent Price as the y-axis, with a linear regression line to determine the relationship. The color rises from green to yellow according to the value of the rent price (ggplot2, n.d.).

Plot:

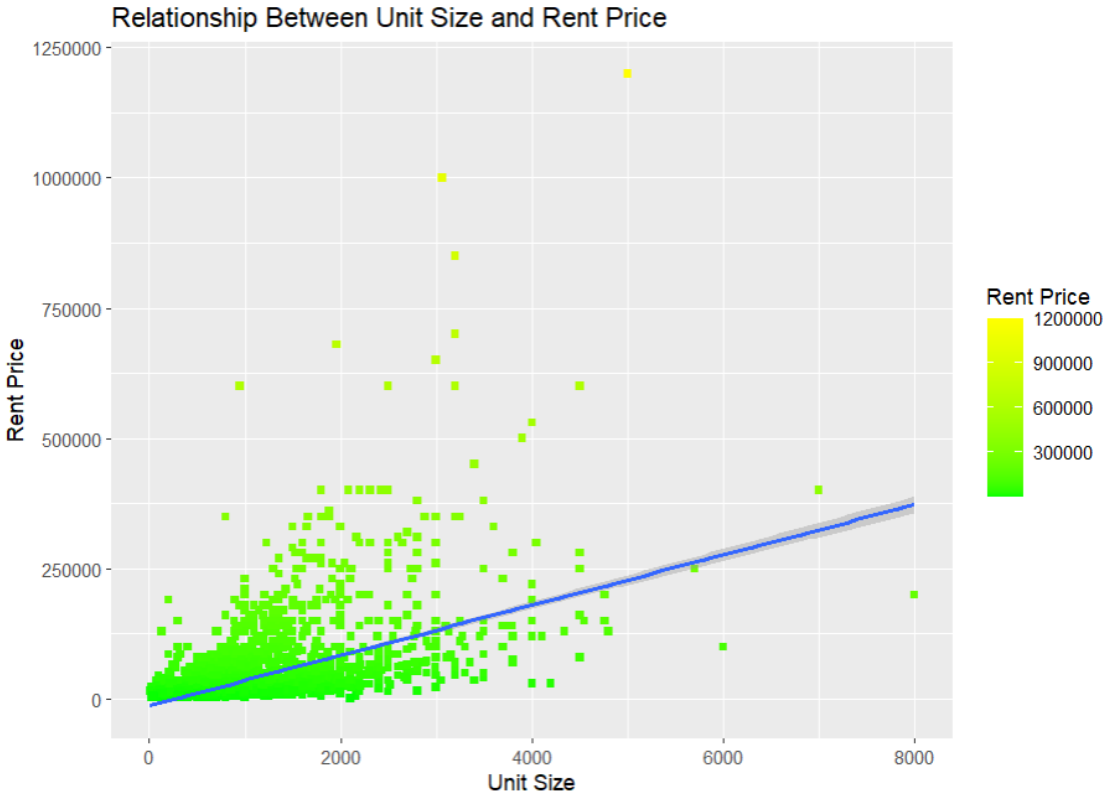

Figure 4-29: Relationship Between Unit Size and Rent Price

According to the plot, the rent price gradually increases as the unit size increases. Hence, the unit size contributes to one of the factors of affecting the rent price. The bigger the unit, the more the unit is worth due to its bigger space and the materials and resources used to build the unit.

4.2.2 Analysis 2-2: Relationship between BHK and rent price

Analysis techniques used:

Analysis Technique	Reason
Data Visualization	A bar plot is plotted to determine the distribution of rent price according to BHK In Unit

Source Code:

```
#Analysis 2-2 : How does the BHK affect the rent price?

bhk_price <- house_data %>%
  group_by(`BHK In Unit`) %>%
  summarise(bathroom_mean=round(mean(`Rent Price`)), 2)

ggplot(bhk_price, aes(x=`BHK In Unit`, y= bathroom_mean)) +
  geom_bar(aes(fill=`BHK In Unit`), stat = "identity") +
  geom_text(aes(label=bathroom_mean)) +
  labs(title = "BHK In Unit vs Rent Price of Houses for Rental", x="BHK In Unit", y="Price")
```

Figure 4-30: Visualize the Distribution of Rent Price According to BHK In Unit

A bar chart is plotted with BHK In Unit as the x-axis and Rent Price as the y-axis, with the color of the bars distinguished with the categories present within the BHK In Unit column. The stat is set to identity so that the heights of the bars is able to represent to values within the data.

Plot:

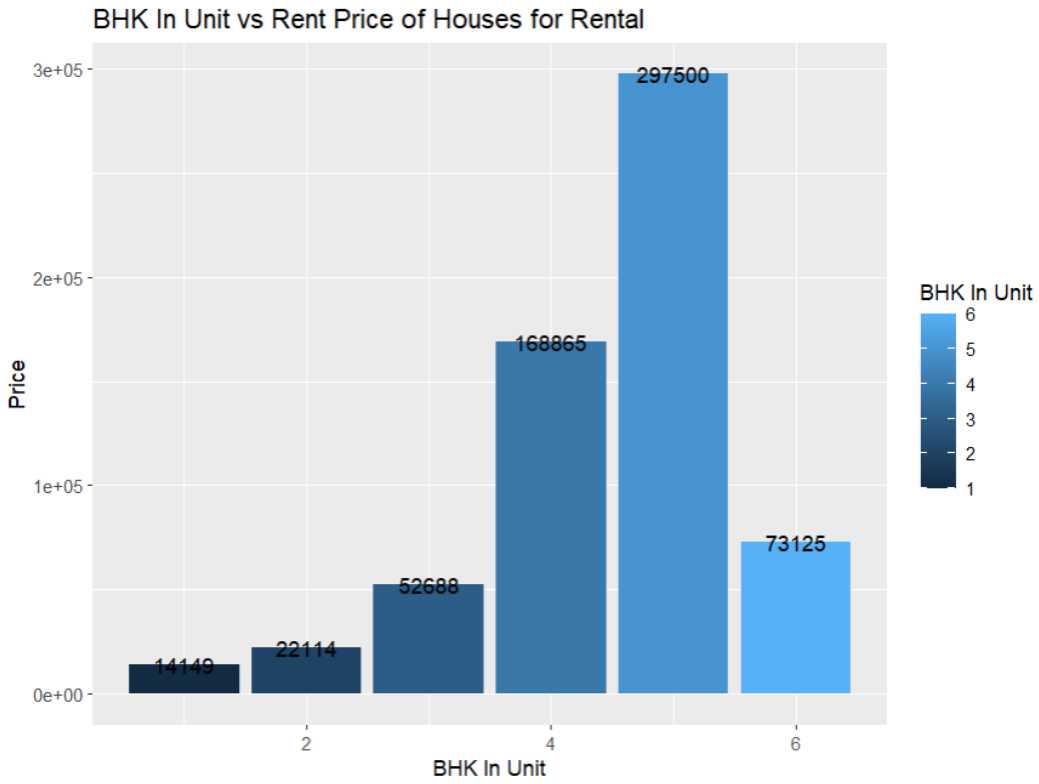

Figure 4-31: BHK In Unit vs Rent Price of Houses for Rental

The bar chart states that the mean price gradually increases and reaches its peak at 5 BHK, then its decline is significant after its peak. This could be related to the cost spent on construction and architectural refinements made for the house as big houses requires a higher cost for maintenance.

4.2.3 Analysis 2-3: Mean of the rental price for each type of areas

Analysis techniques used:

Analysis Technique	**Reason**
Data Manipulation	The rounded mean of the rent price for each area type is aggregated
Data Visualization	A bar chart is plotted according to the rent price for each area type

Source Code:

```
#Analysis 2-3: Mean of the rental price for each type of areas.

mean_area <- function(area)
{
  round(mean(house_data[house_data$`Type of Area` == area, 'Rent Price']), 2)
}

mean_super = mean_area("super")
mean_carpet = mean_area("carpet")
mean_built = mean_area("built")

#Construct bar chart to compare mean for each area
mean_area <- c(mean_super, mean_carpet, mean_built)
mean_area_label <- c("Super Area", "Carpet Area", "Built Area")


mean_area_plot <- barplot(mean_area, main = "Distribution of Mean for Types of Area", xlab = "Type of Area",
                          ylab = "Mean Value", names.arg = mean_area_label, col = mean_area, ylim = c(0,70000))
text(mean_area_plot,0, mean_area, cex = 1, pos = 3)
```

Figure 4-32: Calculate and Visualize the Rental Price of Each Type of Areas

The mean_area() function is defined to calculate the mean of the rent price according to the area type rounded into 2 decimal places. The function is called for the mean_super, mean_carpet and mean_built variables along with passing the arguments – super, carpet and built respectively in the form of strings. Then, a bar chart is plotted according to the outcome of the variables, which is stored in a vector.

Plot:

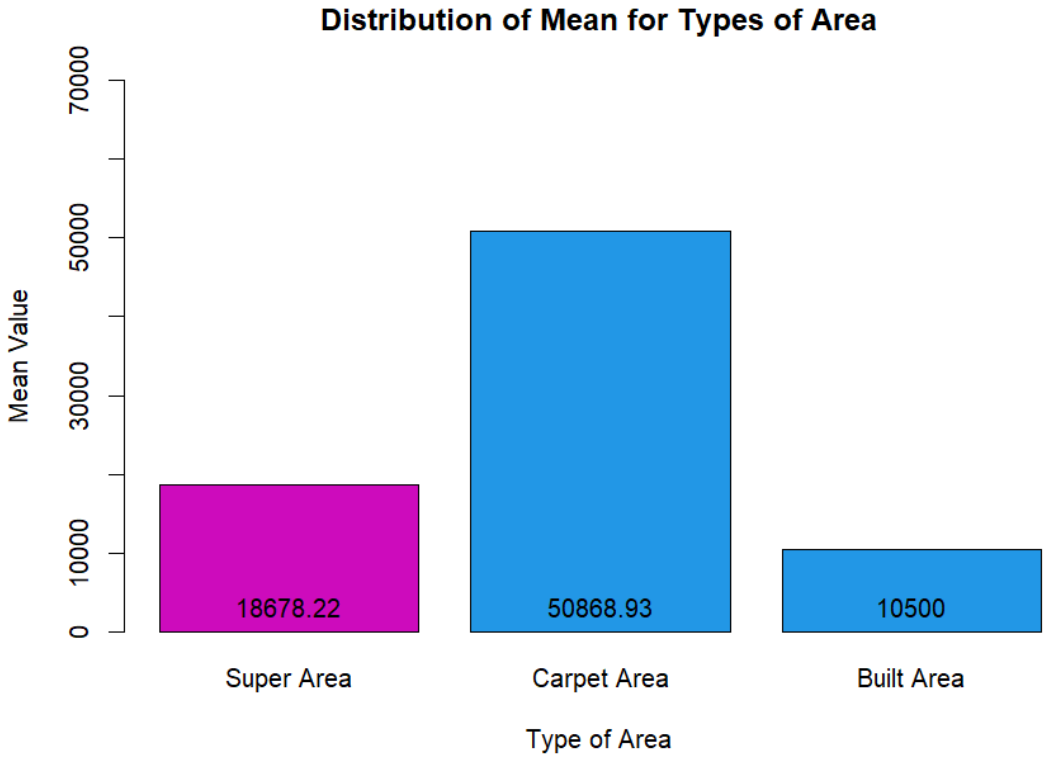

Figure 4-33: Distribution of Mean Rent Price of Types of Area

The carpet area tends to have the highest number of mean rent price as it is worth 50868.93 INR, approximately 60% higher than super area and built area. The average rent price for carpet areas is higher as most high-priced houses falls into the carpet area category. Builders might also use the built area to represent the carpet area calculation to obtain more profit from the lack of clarity calculation of space as it can lower the cost of per square feet of the property (Mishra, 2022). Hence, providing a false impression for people that the houses are invested in an extremely reasonable price. Other than that, the areas represent different types of equations used to calculate the house price, while varying the house rent price as well.

4.2.4 Analysis 2-4: Mean of the rental price for each type of furnishings

Analysis techniques used:

Analysis Technique	Reason
Data Manipulation	The rounded mean of the rent price for each area type is aggregated
Data Visualization	A bar chart is plotted according to the rent price for each area type

Source Code:

```
#Analysis 2-4: Mean of the rental price for each type of furnishings.

mean_furnishings <- function(furnishings)
{
  round(mean(house_data[house_data$Furnishings == furnishings, 'Rent Price']), 2)
}

mean_unfurnished = mean_furnishings("unfurnished")
mean_semi_furnished = mean_furnishings("semi-furnished")
mean_furnished = mean_furnishings("furnished")

#Construct bar chart to compare mean of each furnishings
mean_furnishing <- c(mean_unfurnished, mean_semi_furnished, mean_furnished)
mean_furnishing_label <- c("Unfurnished", "Semi-furnished", "Furnished")

mean_furnishing_plot <- barplot(mean_furnishing, main = "Distribution of Mean Rent Price for Types of Furnishing", xlab = "Types of Furnishing"
                                , ylab= "Mean Value", names.arg = mean_furnishing_label, col = mean_furnishing, ylim = c(1, 60000))
text(mean_furnishing_plot,0, mean_furnishing, cex = 1, pos = 3)
```

Figure 4-34: Calculate and Visualize the Rental Price of Each Type of Furnishing

The mean_furnishings() function is defined to calculate the mean of the rent price according to the type of furnishings rounded into 2 decimal places. The function is called for the mean_unfurnished, mean_semi_furnished and mean_furnished variables along with passing the arguments – unfurnished, semi-furnished and furnished respectively in the form of strings. Then, a bar chart is plotted with mean_furnishing as the x-axis and the values as the y-axis.

Plot:

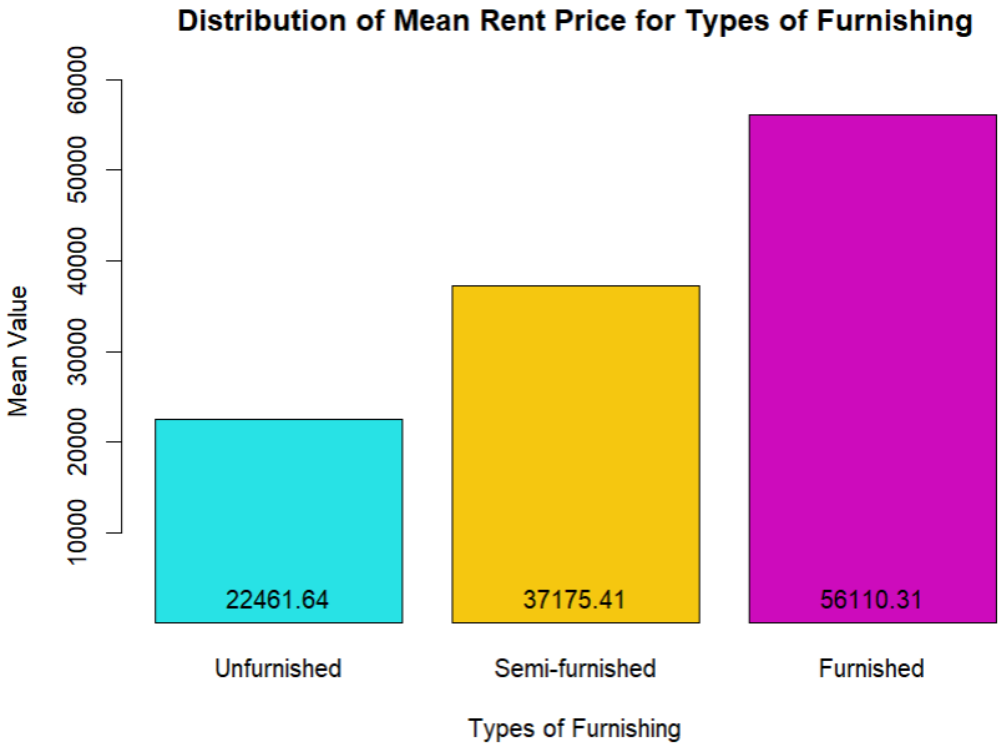

Figure 4-35: Distribution of Mean Rent Price of Types of Furnishing

According to the bar plot, furnished units are sold in the highest mean rent price compared to unfurnished and semi-furnished units. The high price could be due to the occurrence of depreciation to assets such as tables, couches and chairs and to cover the cost of small utensils such as toiletries, dish soap and floor mats.

4.2.5 Analysis 2-5: Relationship between the number of bathrooms and the rent price

Analysis techniques used:

Analysis Technique	Reason
Data Visualization	A bar plot is plotted to determine the distribution of rent price according to the number of bathrooms

Source Code:

```
#Analysis 2-5: Relationship between the number of bathrooms and the rent price.

bathroom_price <- house_data %>%
  group_by(`Number of Bathroom`) %>%
  summarise(bathroom_mean=round(mean(`Rent Price`)), 2)

ggplot(bathroom_price, aes(x=`Number of Bathroom`, y= bathroom_mean)) +
  geom_bar(stat = "identity", aes(fill=`Number of Bathroom`)) +
  geom_text(aes(label=bathroom_mean), position = position_stack(vjust = 0.5))
  labs(title = "Number of Bathrooms vs Rent Price of Houses for Rental", x="Number of Bathroom", y="Price")
```

Figure 4-36: Visualize the Distribution of Rent Price According to Number of Bathroom

A bar chart is plotted with Number of Bathroom as the x-axis and the Average Rent Price as the y-axis, with the color of the bars distinguished with the categories present within the Number of Bathroom column. The stat is set to identity so that the heights of the bars can represent to values within the data.

Plot:

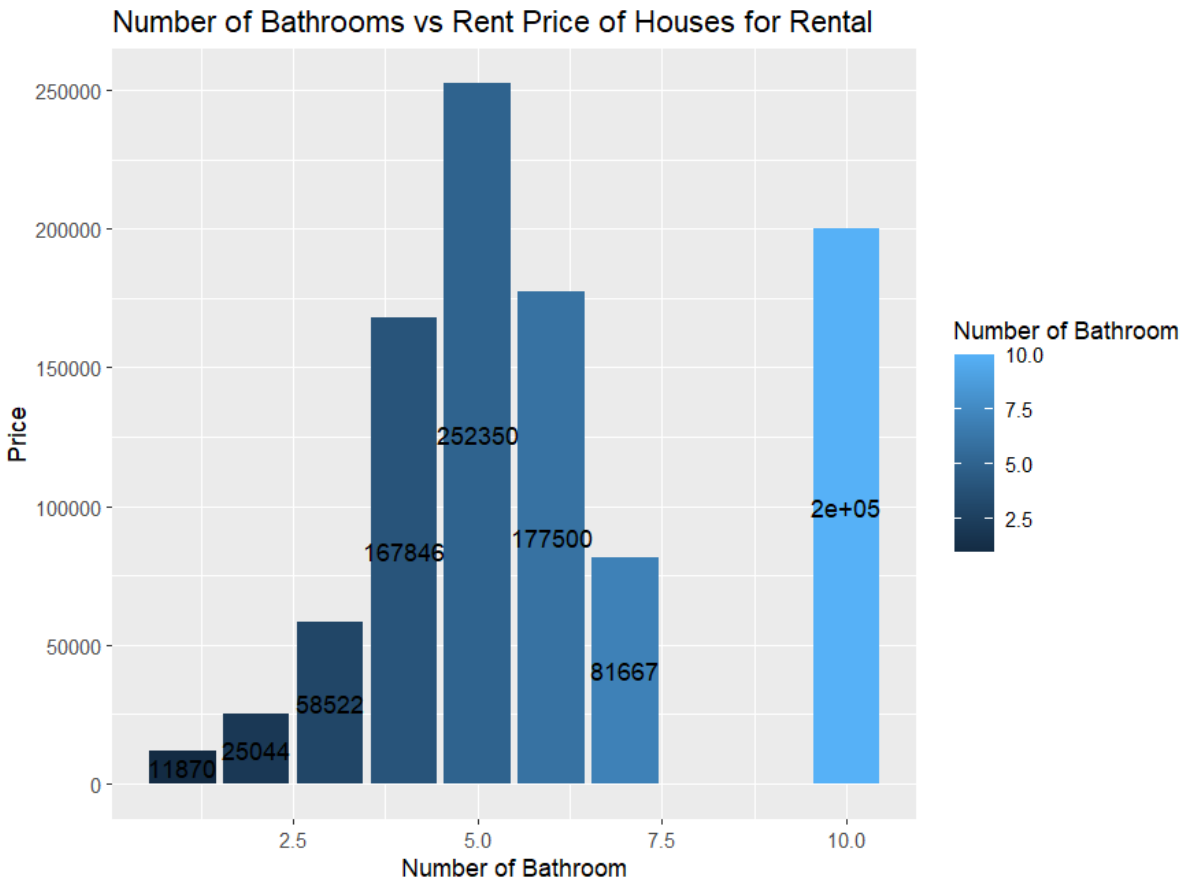

Figure 4-37: Number of Bathroom vs Rent Price of Houses for Rental

The bar chart illustrates that the rent prices increase and reaches its peak at 5 bathrooms. Then, a significant drop can also be deduced after reaching its peak. The rent price according to the number of bathrooms also varies to the BHK of the house, it is also shown that houses with 5 BHK has the highest average rent price.

4.2.6 Analysis 2-6: Relationship between the floor number and rent

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The number of bachelors that prefers different types of building height is explored
Data Manipulation	The mean and median of the total number of floors is calculated and a new column is derived according to the values
Data Visualization	A pie chart relating to the distribution of building height for bachelors are plotted

Source Code:

Data Binning

Extra Feature 6: ntile()

```
#Calculate bin mean
floor_bin_one <- mean(binned_data[binned_data$`Floor Bin` == 1, 'Rent Price'])
floor_bin_two <-mean(binned_data[binned_data$`Floor Bin` == 2, 'Rent Price'])
floor_bin_three <-mean(binned_data[binned_data$`Floor Bin` == 3, 'Rent Price'])
floor_bin_four <-mean(binned_data[binned_data$`Floor Bin` == 4, 'Rent Price'])
floor_bin_five <-mean(binned_data[binned_data$`Floor Bin` == 5, 'Rent Price'])

floor_bin_rent <- c(floor_bin_one, floor_bin_two, floor_bin_three, floor_bin_four, floor_bin_five)
floor_bin <- c(1,2,3,4,5)

floor_bin_data <- data.frame(floor_bin,floor_bin_rent)

ggplot(floor_bin_data, aes(x=floor_bin, y= floor_bin_rent, color=floor_bin_rent)) +
  scale_colour_gradient2(low = "red", mid = "yellow" , high = "seagreen", midpoint=median(floor_bin_rent)) +
  geom_line() + geom_point() +
  labs(title = "Floor vs Rent Price", x="Floor Range", y="Rent Price")
```

Figure 4-38: Bin Floor Number Data and Visualize the Data

After exploration, it is found that the data for the floor numbers are noisy. Hence, binning is done to smoothen the lines for better visualization. The data is binned to 5 bins and the mean rent price of each bin is calculated to obtain the bin means so that the values within the bin can be unified to the mean value to reduce the noisiness of data. With the use of the values, a separate data frame is created to fit the bin number and the bin mean value. A line plot is plotted with floor_bin as the x-axis and floor_bin_rent as the y-axis. The

scale_colour_gradient2() function is used so that decision makers can differentiate and compare the values present.

Plot:

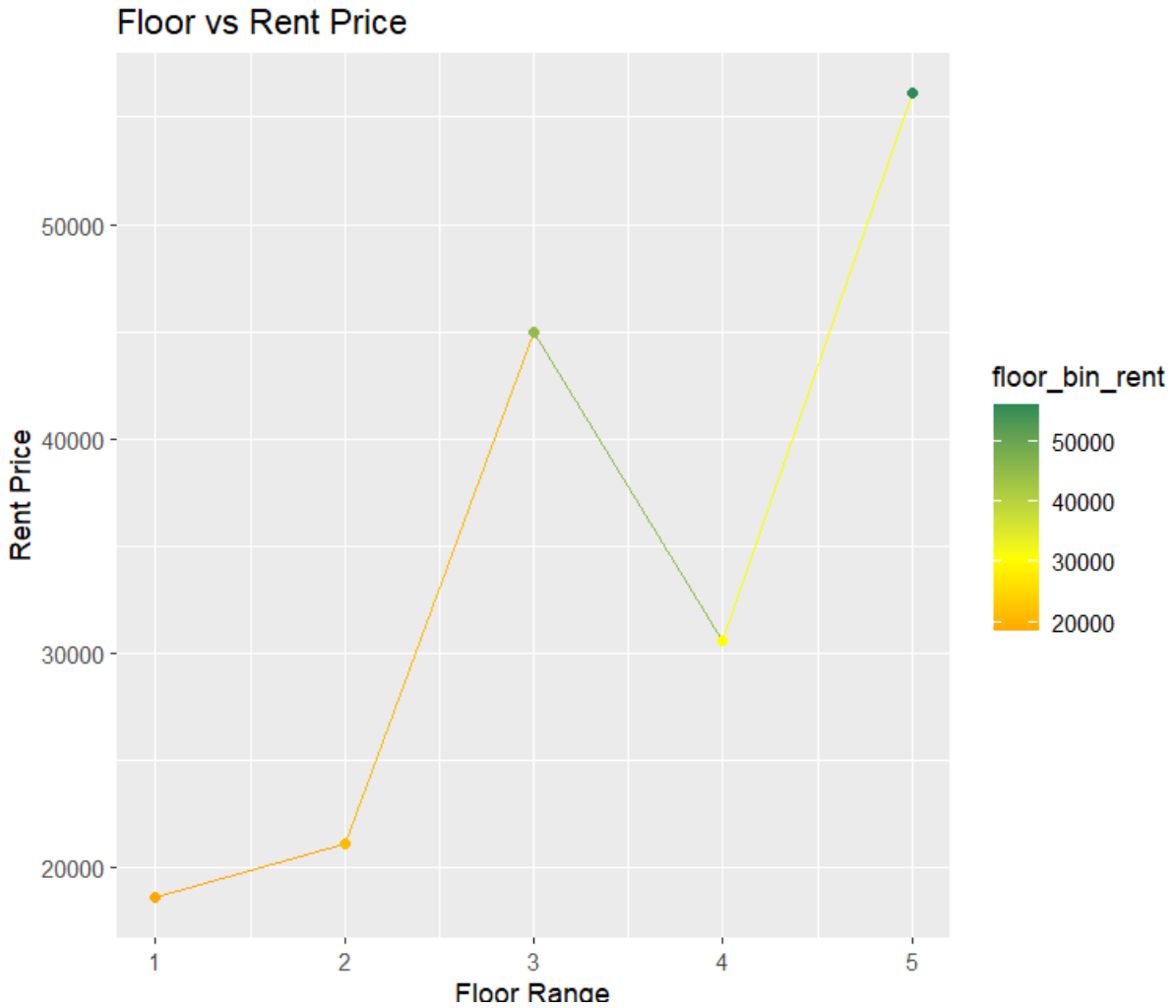

Figure 4-39: Floor vs Rent Price

Based on the line chart, the rent price is the highest on the higher floors. From the lower floors, the price gradually increases until the middle levels where it starts decreasing until the upper middle floors. Top floor units tend to provide a better quality of life as there is more natural light and fresh air, it is also safer in terms of security as its sense of security is heightened as well by creating distance for isolation. The reason of the sudden decrease of prices for the upper middle levels could be due to lesser demand from buyers, and a lower price is established to attract more buyers to purchase.

4.2.7 Analysis 2-7: Relationship between the total number of floors and the rent price

Analysis techniques used:

Analysis Technique	Reason
Data Visualization	A scatterplot is plotted to determine the relationship between the total number of floors and rent price with the help of a linear regression lines

Source Code:

```
#Analysis 2-7: Relationship between the total number of floors and the rent price

ggplot(house_data, aes(x=`Total Number of Floors`, y=`Rent Price`)) +
  geom_point(shape = 15, aes(colour = `Rent Price`)) +
  geom_smooth(method=lm, color = "purple") +
  scale_colour_gradient(low = "pink", high = "blue") +
  labs(title= "Total Number of Floors vs Rent Price")
```

Figure 4-40: Plot Relationship Between the Total Number of Floors

To determine the relationship between the total number of floors and rent price, a scatterplot is plotted with the Total Number of Floors as the x-axis and Rent Price as the y-axis, with a linear regression line to determine the relationship. The color rises from pink to blue according to the value of the rent price.

Plot:

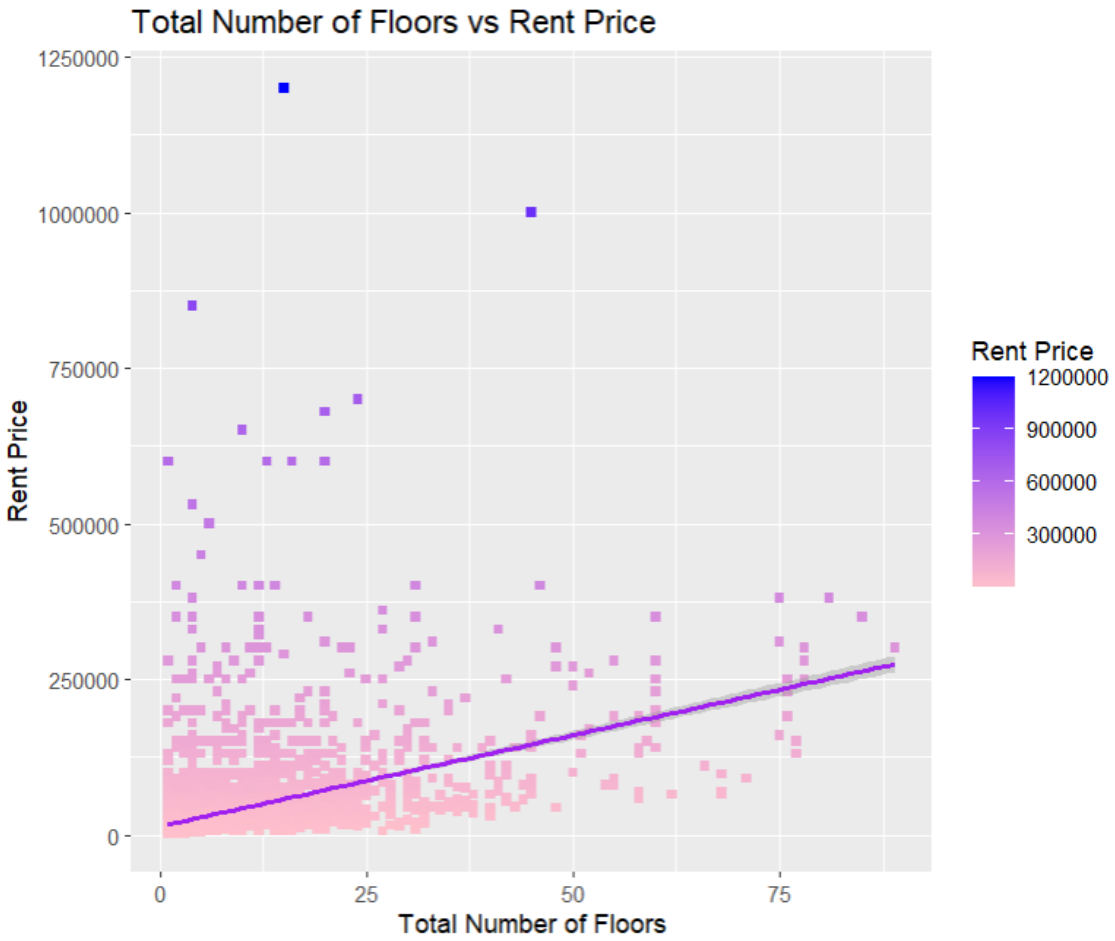

Figure 4-41: Total Number of Floors vs Rent Price

As the total number of floors increases, the rent price increases without taking into account other factors such as furnishings and unit sizes. This could be due to the higher cost used for sophisticated structural engineering efforts, better facilities, and technological improvements. Residents are willing to fork out a big amount to reside within taller buildings due to the high level of convenience in terms of lifestyle (Ahlfeldt & Barr, 2020).

4.2.8 Analysis 2-8: Mean of the rental price for each city

Analysis techniques used:

Analysis Technique	Reason
Data Visualization	A box plot is plotted to determine the range of the rental price for each city

Source Code:

```
#Analysis 2-8: Mean of the rental price for each city.

ggplot(house_data, aes(y=log(`Rent Price`), x=`City Located`, fill=`City Located`)) +
  geom_boxplot() +
  labs(title="Range of Rental Price According to City")
```

Figure 4-42: Visualize the Distribution of Mean Rental Price for Each City

A box plot is plotted as with the logged value of the Rent Price column as the y-axis to properly observe the boxes for better comparison whereas the City Located is the x-axis.

Plot:

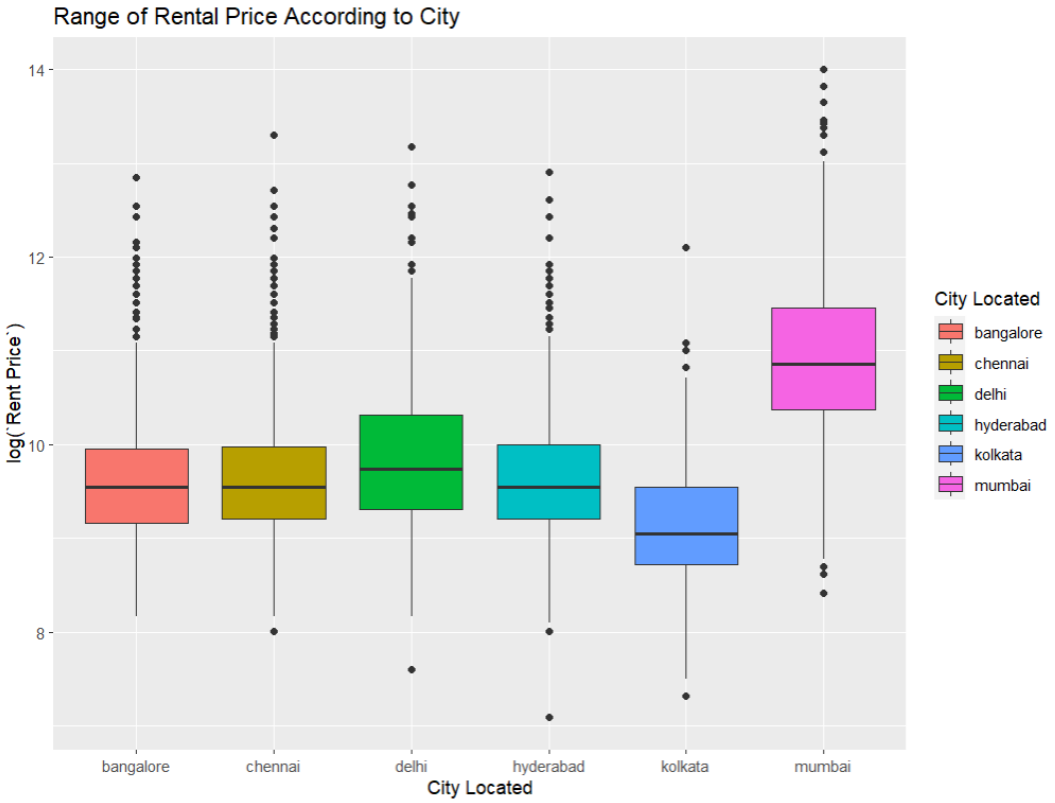

Figure 4-43: City Located vs Rent Price of Houses for Rental

According to the bar plot, Mumbai has the highest rent price among the 6 cities, with the average rent of 85321 INR, approximately 4 times of the average rental present within other cities. The price range in Mumbai is greater than other cities as the prices within the city are more diverse for buyers. The city's environment is catered to the upper-class society where lavish prices are charged for most items and properties. Hence, it is also labelled as the most expensive city to live in India due to its high cost of living. Kolkata has the lowest rent as most people fear to live within that city due to political instability and lesser job opportunities.

4.2.9 Analysis 2-9: Clustering of the pricing of the properties

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	Suitable columns to determine the clusters are explored and selected
Data Manipulation	The values present within the selected columns are scaled
Data Visualization	The cluster plot is visualized to determine the clusters present according to the selected column

Determine Number of Clusters

Extra Feature 9: fviz_nbclust()

K-means Clustering

Extra Feature 7: kmeans() and fviz_cluster()

Source Code:

```
cluster_column <- data.frame(as.numeric(house_data$`Total Number of Floors`),
                             house_data$`Rent Price`)
cluster_column <- scale(cluster_column)
set.seed(123)
```

Figure 4-44: Select and Scale Columns

```
fviz_cluster(model, data = cluster_column)
```

Figure 4-45: Visualize Cluster Plot

The Total Number of Floors and Rent Price columns are selected for the clustering process, the data is scaled with the scale() function as the k-means algorithm should not depend on units of arbitrary variables. The set_seed() function is used to determine a seed for R's random number generator (Datanovia, n.d.). The clusters are then visualized with the fviz_cluster() function by taking the k-means outcome and the initial data as arguments, observations will be represented using points.

Cluster Plot:

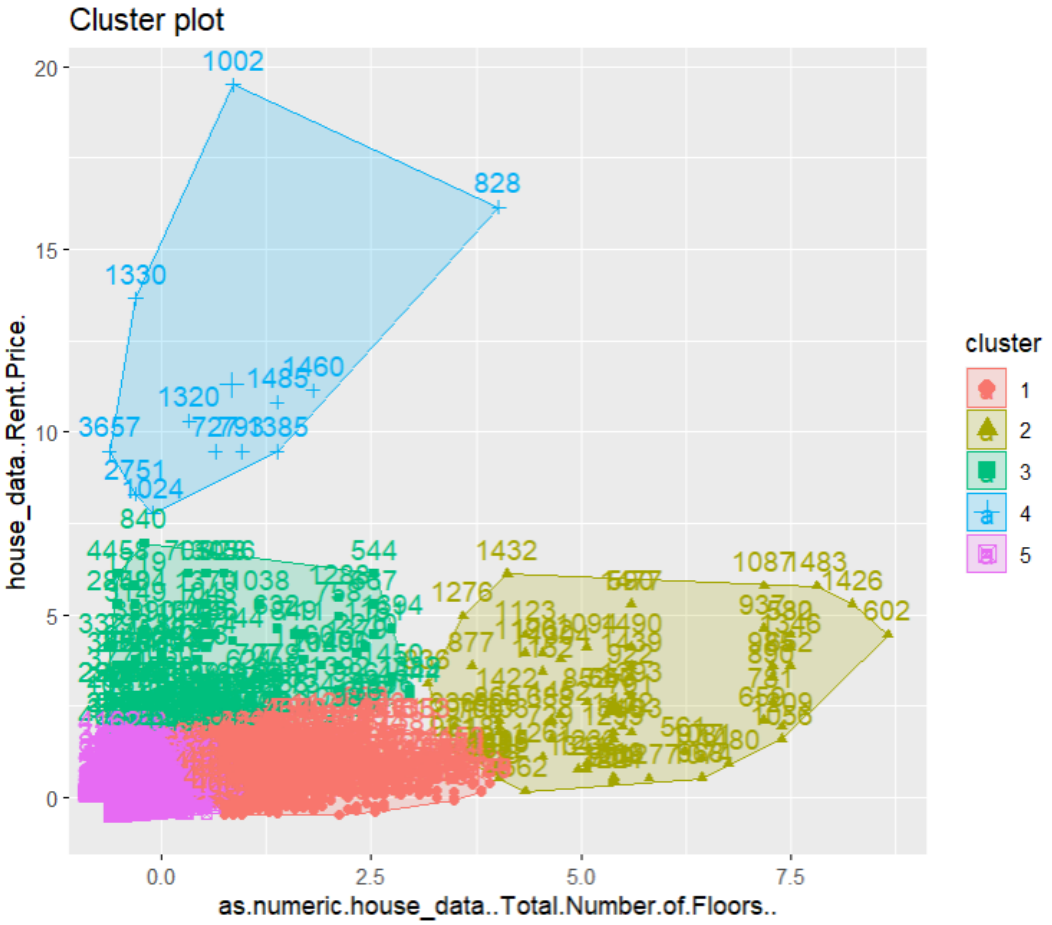

Figure 4-46: Cluster Plot

The data is segmented to 5 clusters, which will be labelled as budget, slightly budget, moderate, slightly expensive and expensive. The clusters are visualized according to the number of clusters determined in the kmeans() function through the elbow method (refer to Extra Feature 7: Clustering).

4.2.10 Analysis 2-10: Distribution of budget and expensive properties according to Analysis 2-9

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	Characteristics of the cluster are examined
Data Manipulation	The clusters are labelled into a new column – Price Category after binding it into the dataset
Data Visualization	The distribution of each price category produced by the cluster is plotted into a bar chart

Source Code:

```
#Analysis 2-10: Distribution of budget and expensive properties according to Analysis 2-9

#Add the clusters as a column
house_data <- cbind(house_data, 'Price Category' = model$cluster)
head(house_data)

#Check the count for each cluster
model$size

#Check cluster means
model$centers

#Change the cluster number to meaningful labels
house_data["Price Category"][house_data$`Price Category` == 1,] <- "Slightly Budget"
house_data["Price Category"][house_data$`Price Category` == 2,] <- "Moderate"
house_data["Price Category"][house_data$`Price Category` == 3,] <- "Slightly Expensive"
house_data["Price Category"][house_data$`Price Category` == 4,] <- "Expensive"
house_data["Price Category"][house_data$`Price Category` == 5,] <- "Budget"

#Plot bar chart
ggplot(house_data, aes(x =`Price Category`)) +
  geom_bar(aes(fill=`Price Category`)) +
  labs(title = "Distributions of Price Categories")
```

Figure 4-47: Exploring, Transforming and Visualizing Cluster Data

The clusters are bound in the form of column as a new column – Price Category. The size and the centers of the clusters are then inspected to gain more relevant insights about the clusters for accurate labels before initiating the labelling process. Based on the results of the cluster centers generated, the clusters are then labelled into 5 categories – budget, slightly budget, moderate, slightly expensive and expensive. A bar chart is plotted to illustrate the distribution of each price category.

Plot:

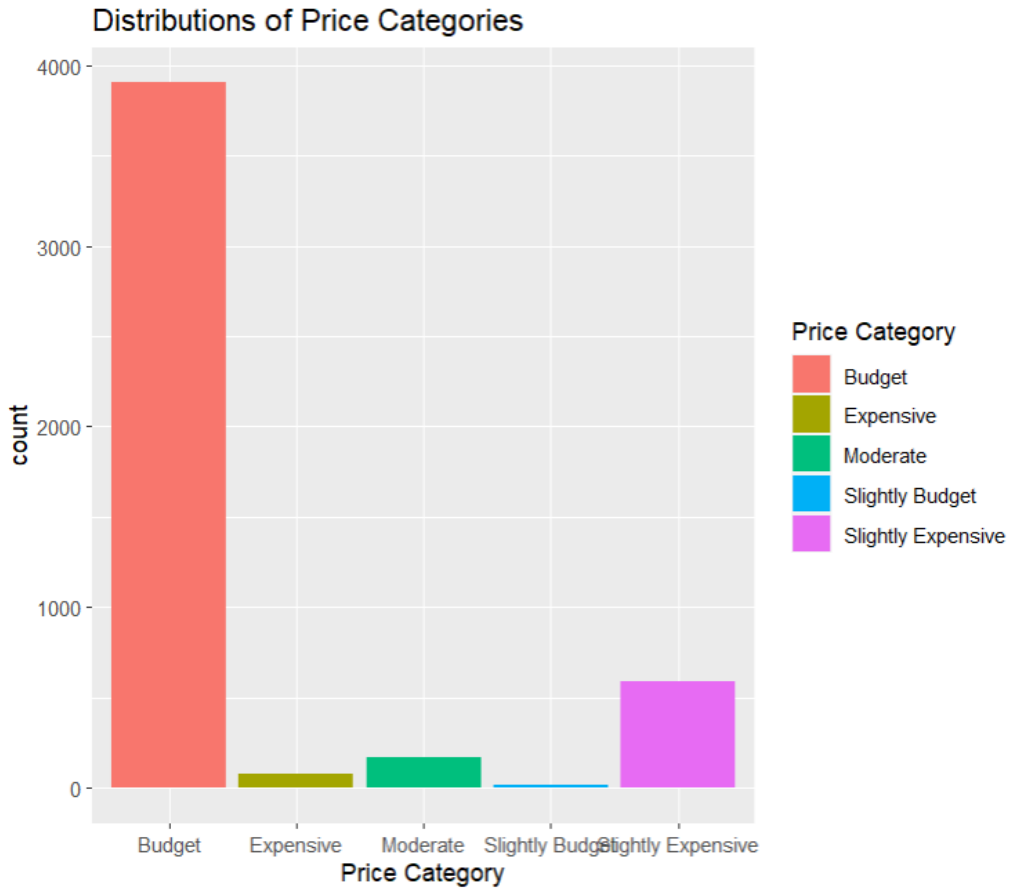

Figure 4-48: Distributions of Price Categories

According to the bar plot, most of the properties present within the data set are categorized as budget units, followed by slightly expensive, moderate, expensive and slightly budget units. The budget houses majorly involve houses which are short and cheap in price as it provides convenience to people in a low and affordable price. Houses that are slightly expensive are usually short buildings but with a slightly high rent price compared to the other houses. The low price of houses could be due to the geographical location, competition, housing environment, history and other related factors.

4.2.11 Conclusion

The rent price of the house can be affected by many factors, which includes the unit size, BHK, floor number, total number of floors present in the building and the city located. The following table describes the aspects that affects the rent price of the houses.

Aspect	**Relationship**
Unit Size	Linear Relationship (Rent Price increases as the Unit Size increases)
BHK	Non-Linear Relationship (The rent price varies upon several factors relating to the BHK, e.g. The demand of the selected BHK
Area Type	The area type affects the rent price as different ways of calculating are used to determine the house price.
Furnishings	The rent price is affected by the number of furnishings present as owners do expect a higher price in return of providing a complete set of furniture and appliances for buyers.
Number of Bathrooms	The rent price varies upon the rent price according to BHK as the number of bathrooms hold a specific relationship with the BHK.
Floor Number	The graph does not indicate any relationships but instead it displays the trend of buyer behavior
Total Number of Floors	Linear Relationship (Rent Price increases as the Total Number of Floors increases)
City Located	There are several factors within the city located that affects the price, which are more qualitative than quantitative. It could be due to political, economic, social, technological and environmental factors.

Table 4-1: Relationship Between Related Aspects and the Rent Price

According to data exploration, the clustering of the total number of floors and the rent price has illustrated groupings that are more significant and unique. Through the relationships established by both variables, clusters are determined so that the data can be grouped to 5

categories – budget, slightly budget, moderate, slightly expensive and expensive. The clustering model is decently fit as the values of between and total sum of squares is 80.8%. Main decision makers can have a better idea on how to manage and choose their desired house according to their financials through the results obtained within the clusters. Budget houses are present in the largest amount. Hence, the houses can be deduced as economically affordable for most people.

4.3 Question 3: Which cities have the most demand in renting property? And why?

4.3.1 Analysis 3-1: Distribution of properties within cities

Analysis techniques used:

Analysis Technique	Reason
Data Exploration	The number of rows of properties for each city is explored
Data Visualization	A bar chart is plotted according to the number of properties present within each city

Source Code:

```
#Question 3: Which cities have the most demand on renting property? And why?

#Analysis 3-1: Distribution of properties within cities

city_distribution <- function(city_name)
{
  nrow(house_data[house_data$`City Located` == city_name ,])
}

property_kolkata = city_distribution("kolkata")
property_mumbai = city_distribution("mumbai")
property_bangalore = city_distribution("bangalore")
property_delhi = city_distribution("delhi")
property_chennai = city_distribution("chennai")
property_hyderabad = city_distribution("hyderabad")

city_property <- c(property_kolkata, property_mumbai, property_bangalore, property_delhi,
                   property_chennai, property_hyderabad)
city_property_label <- c("Kolkata", "Mumbai", "Bangalore", "Delhi", "Chennai", "Hyderabad")

city_property_plot <- barplot(city_property, main = "Distribution of Properties Within Cities", xlab = "City"
                              , ylab= "Number of Properties", names.arg = city_property_label,
                            col = city_property, ylim = c(0, 1000))
text(city_property_plot,0, city_property, cex = 1, pos = 3)
```

Figure 4-49: Determine the Distribution of Properties Within Cities

A function named city_distribution() is defined to calculate the number of rows present within the cities selected that is present in the condition. The function is then used on to determine the value of the variables – property_kolkata, property_mumbai, property_bangalore, property_delhi, property_chennai and property_hyderabad by passing on the string values of the city names to their respective variables. The values obtained within the variables are listed as a vector. A bar plot is plotted according to the vector values as the x-axis and the labels of the city name to assist in properly labelling the city names.

Plot:

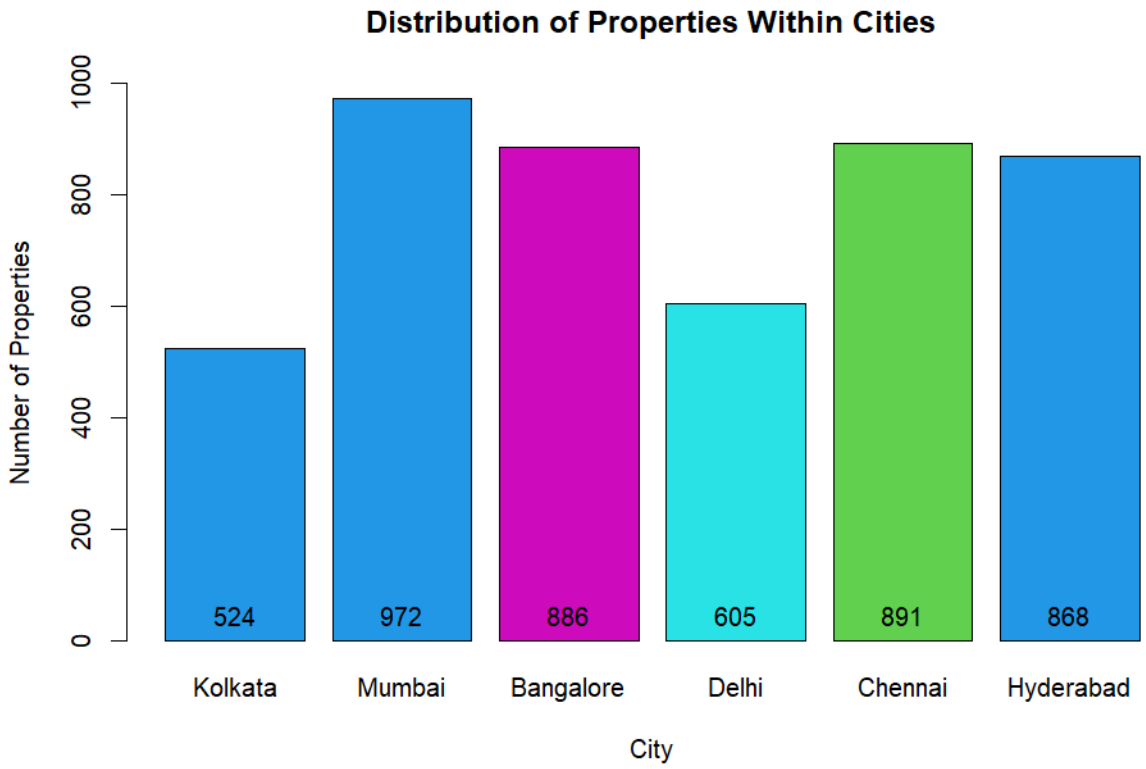

Figure 4-50: Distribution of Properties Within Cities

The bar plot has stated that Mumbai has the highest demand of properties, with the total number of 972 properties. On the other hand, Kolkata has the least demand of properties as it only consists of 524 properties. Most of the buyers prefers to find houses in Mumbai as it is considered as one of the most developed and modern cities within India. Hence, the job opportunities and facilities present within the city is skyrocketing and improving constantly, despite having a higher cost of living compared to other cities. Kolkata has the least demand as it is developing backwards due to the stagnant mindset of residence and the lack of job opportunities, basic amenities and political conflicts within the city. Although the cost of living is lower, buyers do not prefer to reside in the city as it is more prone to crimes and safety hazards.

4.3.2 Analysis 3-2: Distributions of BHK according to cities

Analysis techniques used:

Analysis Technique	Reason
Data Visualization	A bar plot is plotted to determine the distribution of BHK according to cities

Source Code:

```
#Analysis 3-2: (Related to the preference of people) Distributions of BHK according to cities

ggplot(house_data, aes(x = `BHK In Unit`)) +
  geom_bar(aes(fill=factor(`BHK In Unit`))) +
  facet_wrap(~`City Located`) +
  labs(title = "Distributions of BHK According to Cities")
```

Figure 4-51: Visualize the Distribution of BHK According to Cities

A bar plot is plotted to visualize the count of each BHK present within each city. Through utilizing the facet_wrap() function, the bar plots of each BHK will be plotted into separate plots, which will be labelled according to the city name.

Plot:

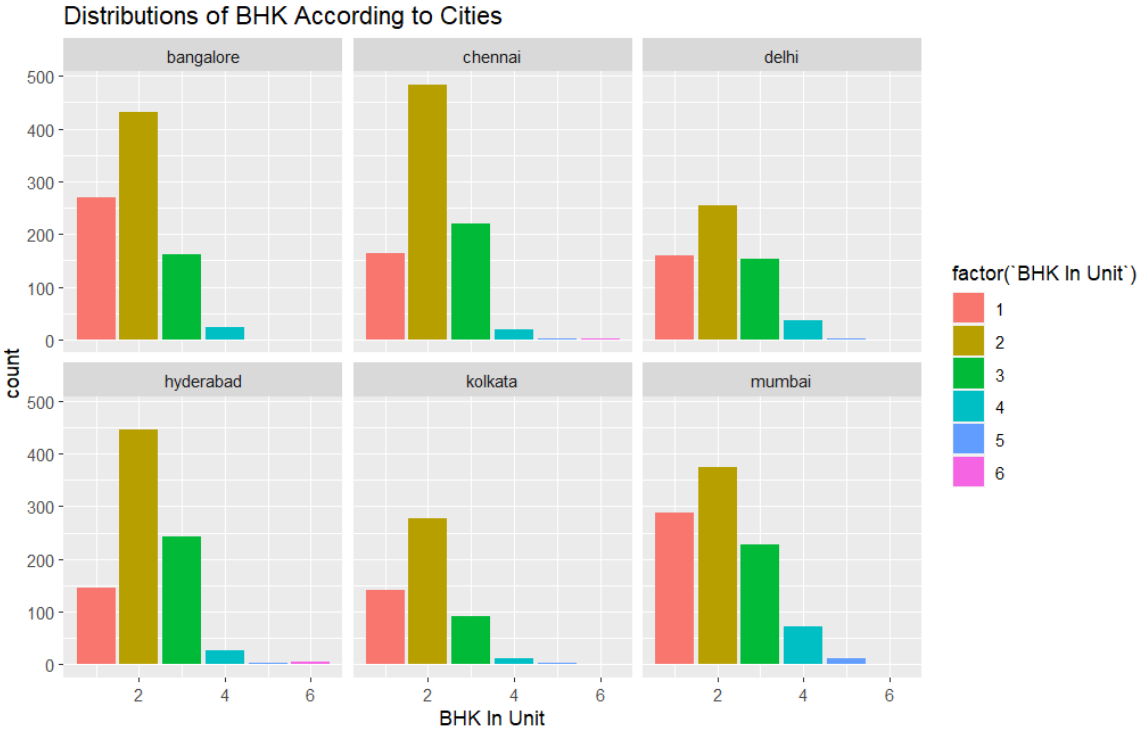

Figure 4-52: Distribution of BHK According to Cities

By comparing to bar plots of respective cities, it can be deduced that Mumbai has the most demand of 1 BHK, 4 BHK and 5 BHK units, Chennai for 2 BHK, Hyderabad for 3 BHK and 6 BHK units. More people might prefer to reside in Mumbai due to its wide range of house choices in terms of BHK and a more rapid advancement. Delhi and Kolkata seem to have lesser varieties of houses in terms of BHK, hence, it might also contribute to the factor of having lesser demand and preference from people. However, 2 BHK houses are the most in demand for each city as both bachelors and families prefer to reside in 2 BHK houses according to previous analysis.

4.3.3 Analysis 3-3: Distributions of areas according to cities

Analysis techniques used:

Analysis Technique	Reason
Data Visualization	A bar plot is plotted to determine the distribution of area types according to cities

Source Code:

```
#Analysis 3-3: (Related to the preference of people) Distributions of areas according to cities

ggplot(house_data, aes(x = `Type of Area`)) +
  geom_bar(aes(fill=factor(`Type of Area`))) +
  facet_wrap(~`City Located`) +
  labs(title = "Distributions of The Type of Areas According to Cities")
```

Figure 4-53: Visualize the Distribution of Area Types According to Cities

A bar plot is plotted to visualize the count of each type of area present within each city. By utilizing the facet_wrap() function, the bar plots of each types of area will be plotted into separate plots, which will be labelled according to the city name.

Plot:

Figure 4-54: Distributions of the Type of Areas According to Cities

According to the bar plots, the presence of carpet areas tends to be the most in Mumbai. Hyderabad has the greatest number of super areas, which is the type of area calculation that people are most fond of. Mumbai's high presence of carpet area calculation might also contribute to the high valuation of houses in terms of rent as housing developers might mark up the pricing for more profit due to the measurement providing only 70% of the total area, which will decrease the pricing of the house. On the other hand, built areas have no to least presence within each city as it is not a popular type of calculation method among people as it includes unnecessary areas such as the thickness of walls.

4.3.4 Analysis 3-4: Distributions of unit sizes according to cities

Analysis techniques used:

Analysis Technique	Reason
Data Visualization	A bar plot is plotted to determine the distribution of unit sizes according to cities

Source Code:

```
#Analysis 3-4: (Related to the preference of people) Distributions of unit sizes according to cities

  #Classify unit sizes to categories
  unit_median <- median(house_data$`Unit Size`)
  house_data <- house_data %>%
    mutate("Unit Size Category"= if_else(house_data$`Unit Size`<= unit_median, 'small unit',
                                         'big unit'))

  ggplot(house_data, aes(x = `Unit Size Category`)) +
    geom_bar(aes(fill=factor(`Unit Size Category`))) +
    facet_wrap(~`City Located`) +
    labs(title = "Distributions of Unit Sizes According to Cities")
```

Figure 4-55: Visualize the Distributions of Unit Sizes According to Cities

Firstly, the median is calculated to determine the splitting point of each value present within the unit size column to form a new column named Unit Size Category. If the unit size is lesser than the median, it will be categorized as a small unit whereas units larger than the median is categorized as a big unit. A bar plot is created to visualize the count of each category of unit size present within each city. By utilizing the facet_wrap() function, the bar plots of each category of unit size will be plotted into separate plots, which will be labelled according to the city name.

Plot:

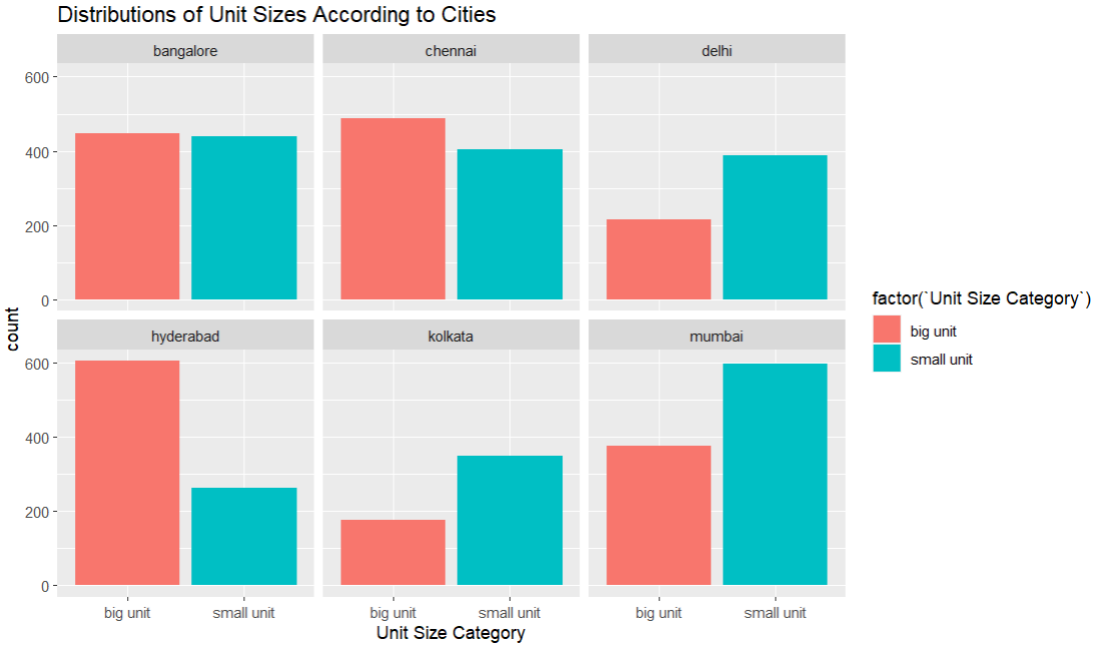

Figure 4-56: Distributions of Unit Sizes According to Cities

Mumbai is the city with most demand as it consists of many smaller units compared to other cities. Based on the previous insights, small units are more popular as it is more affordable and is usually located in well-connected areas over big units that are usually located in more rural areas which provides more trouble in accessing to main city areas. Hence, making Hyderabad a city that is not popular to live in. Bachelors are attracted to living in Bangalore due to the equal amount of unit sizes present within the city, while providing them with more options and flexibility on selecting their houses to rent.

4.3.5 Analysis 3-5: Distributions of furnishings according to cities

Analysis techniques used:

Analysis Technique	Reason
Data Visualization	A bar plot is plotted to determine the distribution of furnishings according to cities

Source Code:

```
#Analysis 3-5: (Related to the preference of people) Distributions of furnishings according to cities

ggplot(house_data, aes(x = Furnishings)) +
  geom_bar(aes(fill=Furnishings)) +
  facet_wrap(~`City Located`) +
  labs(title = "Distributions of The Type of Furnishings According to Cities")
```

Figure 4-57: Visualize the Distribution of Furnishings According to Cities

A bar plot is plotted to visualize the count of each type of furnishings present within each city. By utilizing the facet_wrap() function, the bar plots of each types of furnishings will be plotted into separate plots, which will be labelled according to the city name.

Plot:

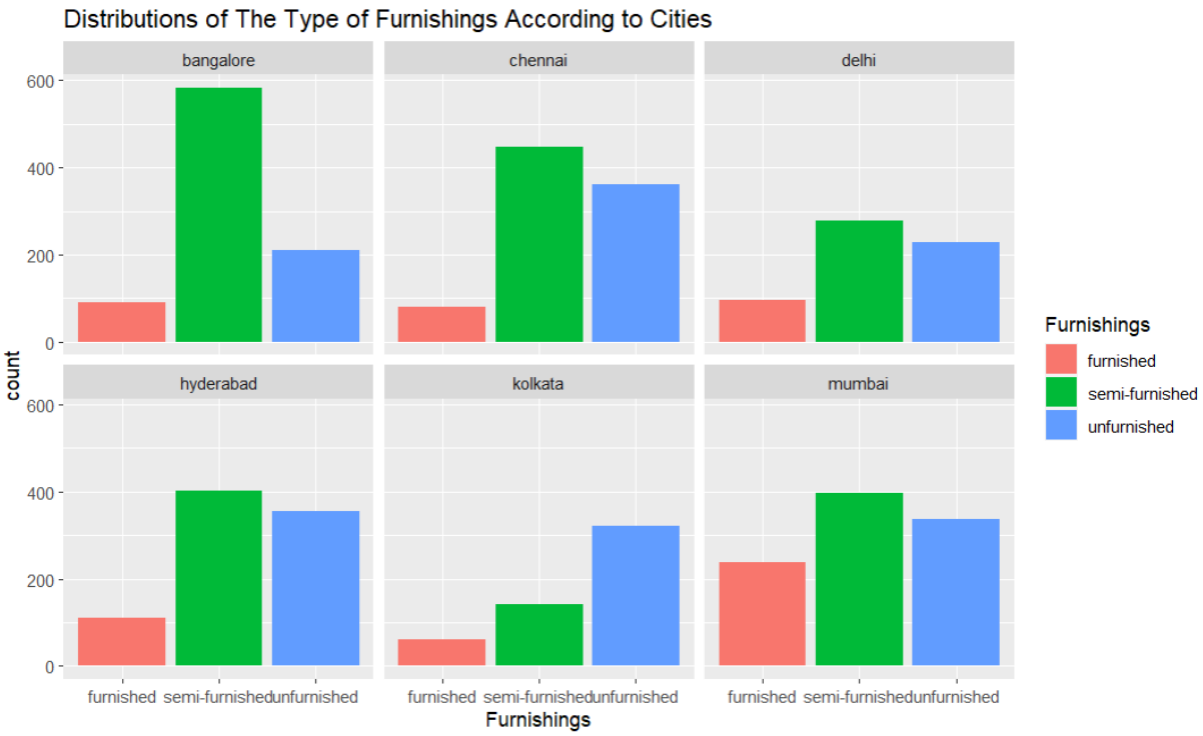

Figure 4-58: Distributions of the Type of Furnishings According to Cities

According to the bar plots, most of the semi-furnished units are located at Bangalore, where it suits majority of the bachelors' preference of renting semi-furnished units. Mumbai consists of the greatest number of furnished units whereas Hyderabad has the most unfurnished units. However, families tend to prefer the semi-furnished units located in Mumbai, as it suits up to their living conditions of a better environment and societal structure. This graph can also support another statement on why Kolkata is not a popular city to live in, as it consists of the least number of semi-furnished units for people of all demographics.

4.3.6 Analysis 3-6: Distributions of the number of bathrooms according to cities

Analysis techniques used:

Analysis Technique	Reason
Data Visualization	A bar plot is plotted to determine the distribution of the number of bathrooms according to cities

Source Code:

```
#Analysis 3-6: (Related to the preference of people) Distributions of the number of bathrooms according to cities

ggplot(house_data, aes(x =as.character(`Number of Bathroom`))) +
  geom_bar(aes(fill=`Number of Bathroom`)) +
  facet_wrap(~`City Located`) +
  labs(title = "Distributions of The Number of Bathrooms According to Cities")
```

Figure 4-59: Visualize the Distributions of the Number of Bathrooms According to Cities

A bar plot is plotted to visualize the count of each number of bathroom present within each city. By utilizing the facet_wrap() function, the bar plots of each number of bathroom will be plotted into separate plots, which will be labelled according to the city name.

Plot:

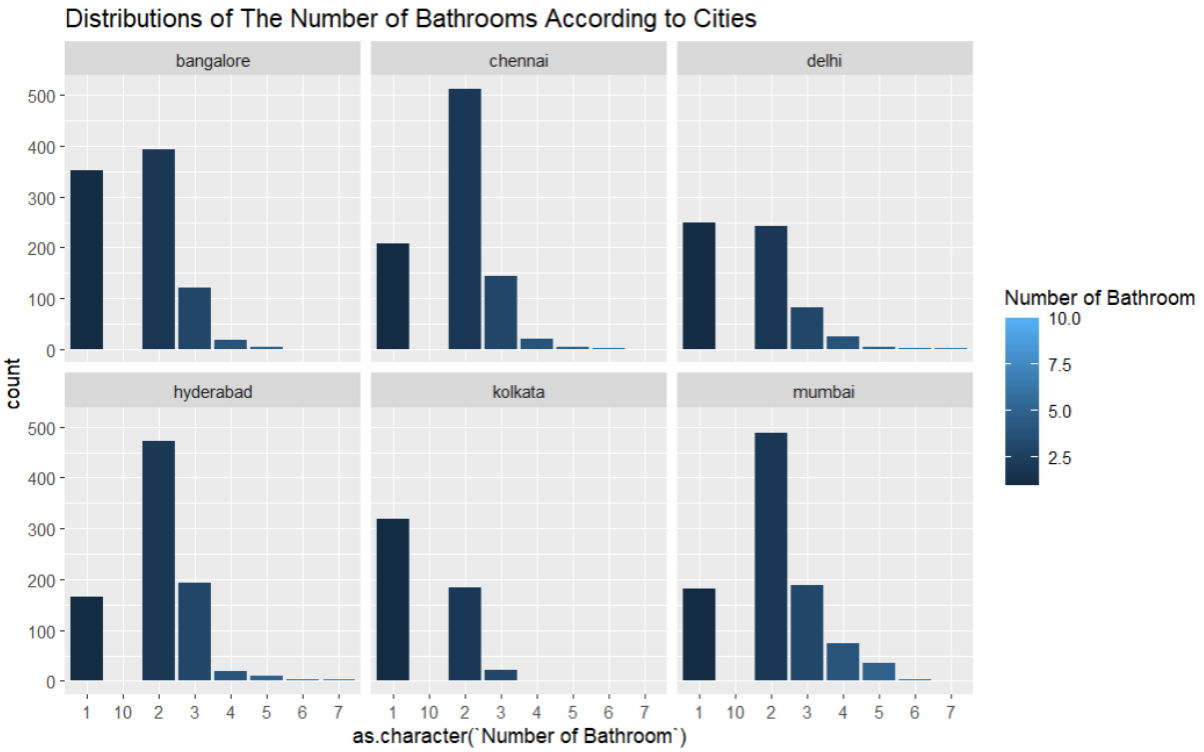

Figure 4-60: Distributions of The Number of Bathrooms According to Cities

Based on the plot above, most of the cities has the highest demand on 2-bathroom units, which is also preferred by most of the buyers. However, cities like Delhi and Kolkata have a higher demand on 1-bathroom units. Both cities share the similarity of having a lesser number of bigger units. Therefore, it can be stated that the presence of 1 bathroom per unit at both cities are high due to the higher presence of smaller units and a lower existence of bigger units compared to other cities, lesser bathrooms have to be built to reduce unnecessary space and facilities for residents to reside in. Other than 1- or 2-bathroom units, units with more than two bathrooms are considered as either luxurious, large or odd, which is usually more friendly for families to move in as they can have a fair share of the bathrooms.

4.3.7 Analysis 3-7: The number of properties and the average rent for each city

Analysis techniques used:

Analysis Technique	Reason
Data Exploration	The number of rows of properties present in each city is explored
Data Manipulation	The mean rental price of each city calculated
Data Visualization	A line plot is drawn to determine the relationship between the number of properties and the price range of each city

Source Code:

```
#Analysis 3-7: The number of properties and the average rent for each city

property_count <- function(city)
{
  nrow(house_data[house_data$`City Located` == city,])
}

average_city_rent <- function(city_name)
{
  mean(house_data[house_data$`City Located` == city_name, 'Rent Price'])
}

kolkata = property_count("kolkata")
bangalore = property_count("bangalore")
chennai = property_count("chennai")
delhi = property_count("delhi")
hyderabad = property_count("hyderabad")
mumbai = property_count("mumbai")

kolkata_rent = average_city_rent("kolkata")
bangalore_rent = average_city_rent("bangalore")
chennai_rent = average_city_rent("chennai")
delhi_rent = average_city_rent("delhi")
hyderabad_rent = average_city_rent("hyderabad")
mumbai_rent = average_city_rent("mumbai")

property_amount <- c(kolkata, bangalore, chennai, delhi, hyderabad, mumbai)
city_name <- c("Kolkata", "Bangalore", "Chennai", "Delhi", "Hyderabad", "Mumbai")
rent_city <- c(kolkata_rent, bangalore_rent, chennai_rent, delhi_rent, hyderabad_rent, mumbai_rent)

property_frame <- data.frame(city_name,property_amount,rent_city)

ggplot(property_frame, aes(x=property_amount, y= rent_city, label=city_name)) +
  geom_point() +
  geom_smooth(method = "loess", color="purple") +
  geom_text(hjust = 0.5, vjust=1.5) +
  labs(title="Relationship Between the Number of Properties and The Average Rent")
```

Figure 4-61: Explore, Calculate and Visualize the Relationship

The property_count() function is defined to calculate the number of rows present for each city to represent the number of properties. Other than that, the average_city_rent() function is created to calculate the mean of the rent price for each city so that the relationship can be described. Each of the functions are assigned to 5 variables respectively to execute their

respective functions and obtain the desired outcome of obtaining the number of properties and the average rent price per city. The variables are then placed in vectors according to their usage and the vectors are combined to construct a data frame. A line graph is then presented in the form of ogives to better illustrate the relationship between the number of properties and the rent price within the cities. The points are labelled with the name of the cities to provide more context for analysis purposes.

Plot:

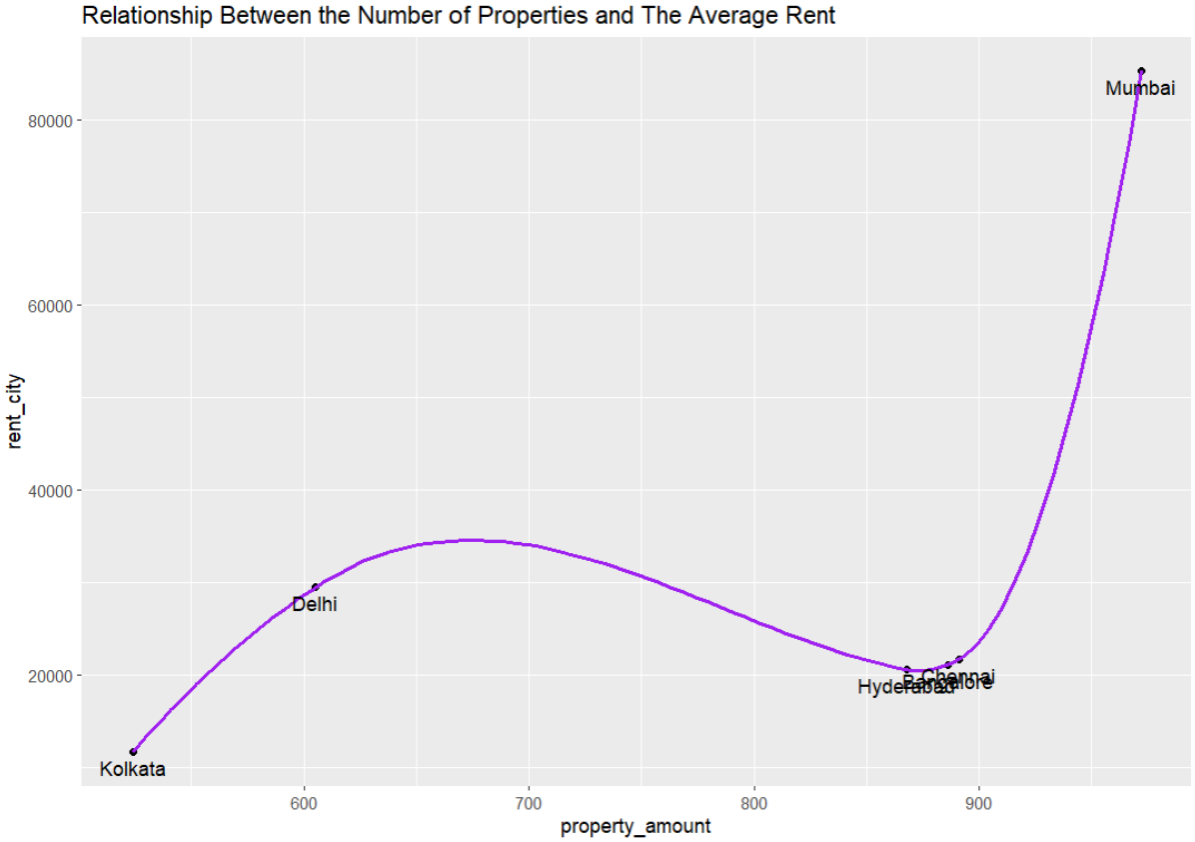

Figure 4-62: Relationship Between the Number of Properties and The Average Rent

Based on the line graph, the mean rent price reached its peak when the number of properties is at its highest. From the starting point of the graph, the mean rent price increases as the property amount increases. However, the mean rent value has dropped when the number of properties is approximately 900, there are three cities who have experienced such scenarios. Through having similar number of properties between the three cities - Hyderabad, Bangalore and Chennai, the pricing is toned down as the competition present to attract suitable buyers are dense and the rent prices are usually the main factor of attraction for buyers.

4.3.8 Conclusion

Mumbai has the most demand of renting property, as it has a large variety of properties with various BHKs for buyers to choose from, which includes 2 BHK houses that deemed as peoples' favorites. In terms of area calculations, Mumbai does not contain the greatest number of super areas but carpet areas, the carpet area calculation has increased the valuation of the houses, but the area involved is still reasonable for most individuals as it does not cover unnecessary areas such as the thickness of walls and other unused facilities within the house. Other than that, Mumbai has also contained the highest number of small units. Therefore, it suits the needs for most people as they prefer to have smaller houses despite either living alone or with their family. Smaller units located in Mumbai provides people with a higher level of comfort if it is well-kept and a more reasonable price range for people to invest in. Buyers have also taken smaller units as more practical compared with larger units where there will be a higher probability of not fully utilizing the space but has to pay a higher price instead.

In terms of furnishings, although Bangalore has more semi-furnished properties compared to Mumbai, which has also become a hotspot for bachelors to rent the houses there, Mumbai consists of a more balanced number of properties with various furnishings for buyers to select and it has a higher number of semi-furnished units as well, which is also aligned to the preferences of families to have semi-furnished units located at Mumbai. The BHK has also affected the number of bathrooms within the unit, Mumbai suits the demands of most people as it contains many units with 2 bathrooms. Although cities like Chennai and Hyderabad have many 2 bathrooms units, the properties there are not popular as people are less likely to prefer the other factors contributed to the unit such as BHK, unit size and furnishings. From the pricing perspective, Mumbai is the most expensive as the number of properties within the city is the most and it has lesser competition with other cities due to its uniqueness in terms of lifestyle, cost of living, landscape, political stability and technological enhancements compared to other cities that falls behind in those aspects.

4.4 Question 4: What is the preference of a family while choosing a house?

4.4.1 Analysis 4-1: Determine the number of families

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The number of families present is explored

Source Code:

```
#Number of Families
number_of_families = nrow(house_data[(house_data$`Preferred Tenants` == "family")
                                     |(house_data$`Preferred Tenants` == "bachelors/family"),])
number_of_families
```

Figure 4-63: Calculate the Number of Families

Outcome:

```
> number_of_families
[1] 3916
>
```

Figure 4-64: Number of Families

Before starting the analysis process, the number of families is calculated to comprehend the number of rows of data relating to families. To calculate the number of families, the number of rows that meets the criteria of having either family or bachelors/family within the Preferred Tenants column is selected for calculation. According to the outcome, there is a total of 3916 families within the dataset.

4.4.2 Analysis 4-2: Determine the preferred BHK by most families

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The distribution of the number of families for each BHK is explored
Data Visualization	The result of the distribution of BHK within families are visualized in the form of bar chart

Source Code:

```
#Analysis 4-1: How many BHK is preferred by most families?
#Distribution of overall bhk unit

family_bhk <- function(bhk)
{
  nrow(house_data[(house_data$`BHK In Unit` == bhk) &
                    ((house_data$`Preferred Tenants` == "family") |
                       (house_data$`Preferred Tenants` == "bachelors/family")),])
}

one_bhk_fam <- family_bhk(1)
two_bhk_fam <- family_bhk(2)
three_bhk_fam <- family_bhk(3)
four_bhk_fam <- family_bhk(4)
five_bhk_fam <- family_bhk(5)
six_bhk_fam <- family_bhk(6)

#Plot barplot to determine the most BHK preferred
bhk_number <- c(one_bhk_fam,two_bhk_fam,three_bhk_fam,four_bhk_fam,five_bhk_fam,six_bhk_fam)
bhk_label <- c("1","2","3","4","5","6")
bhk_family <- barplot(bhk_number, main = "Distribution of BHK In Unit for Families", xlab = "Number of BHK",
              ylab = "Number of Families", names.arg = bhk_label, col = bhk_number, ylim= c(0,2500))
text(bhk_family, 0, bhk_number, cex = 1, pos = 3)
```

Figure 4-65: Calculate and Visualize the Distributions of BHK Within Families

The family_bhk() function is defined to calculate the number of rows for the specified BHK with the condition of having family or bachelors/family as the preferred tenants. The function is called to determine the value of the variables one_bhk_fam, two_bhk_fam, three_bhk_fam, four_bhk_fam, five_bhk_fam and six_bhk_fam with passing arguments according to their respective BHK value. A bar plot is then plotted with bhk_number as the x-axis and the count of each BHK as the y-axis.

Plot:

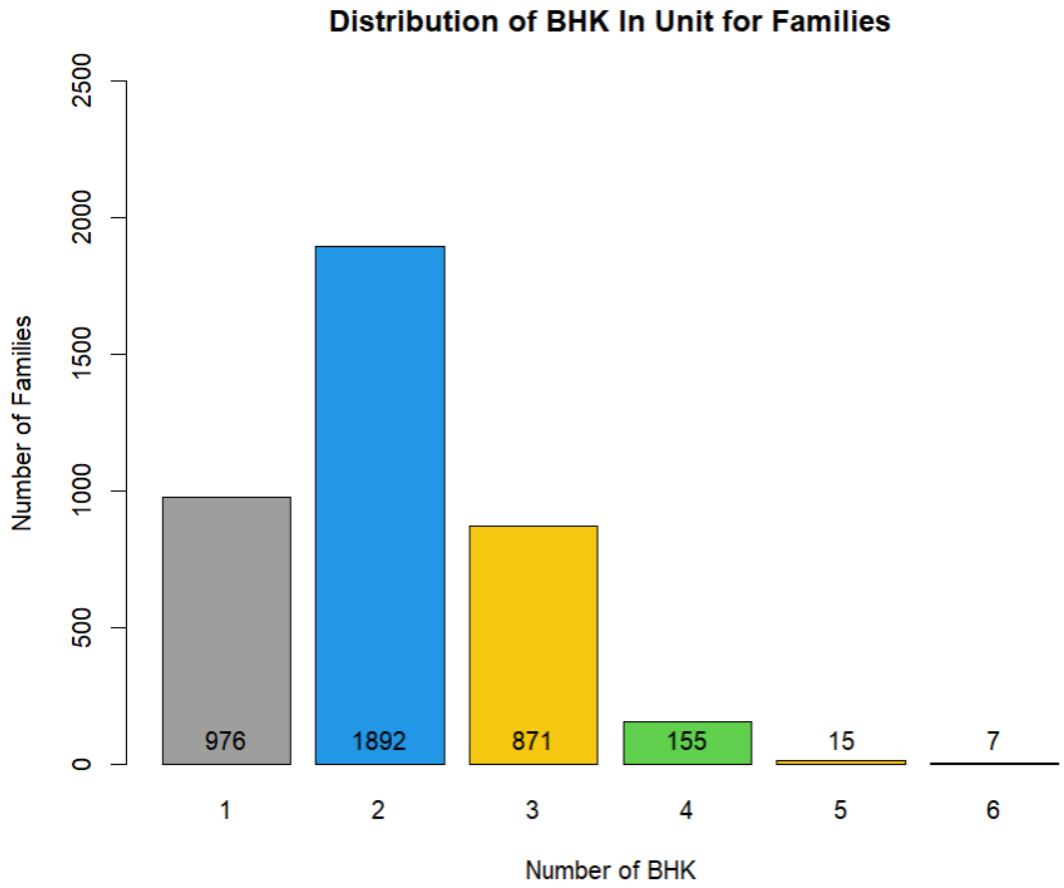

Figure 4-66: Distribution of BHK In Unit for Families

The bar chart states that most families prefer 2 BHK units, followed by 1 BHK and 3 BHK. Families do not prefer to have houses that are 4 BHK and above as most families are small to moderate sized. Hence, it is unnecessary to have houses that consists of too many unused rooms and space and more efforts are required to maintain the house.

4.4.3 Analysis 4-3: Determine the number of bathrooms families prefer

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The distribution of the number of families for each bathroom is explored
Data Visualization	The result of the distribution of the number of bathrooms within families are visualized in the form of circle chart

Source Code:

```
#Analysis 4-2: How many bathrooms do families prefer?
#Distribution of the number of bathrooms

family_bathroom <- function(bathroom)
{
  nrow(house_data[(house_data$`Number of Bathroom` == bathroom) &
                    ((house_data$`Preferred Tenants` == "family") |
                       (house_data$`Preferred Tenants` == "bachelors/family")),])
}

one_bathroom_fam <- family_bathroom(1)
two_bathroom_fam <- family_bathroom(2)
three_bathroom_fam <- family_bathroom(3)
four_bathroom_fam <- family_bathroom(4)
five_bathroom_fam <- family_bathroom(5)
six_bathroom_fam <- family_bathroom(6)
seven_bathroom_fam <- family_bathroom(7)
ten_bathroom_fam <- family_bathroom(10)

#Plot radial plot for bathroom
bathroom_number_fam <- c(one_bathroom_fam, two_bathroom_fam, three_bathroom_fam, four_bathroom_fam,
                         five_bathroom_fam, six_bathroom_fam, seven_bathroom_fam, ten_bathroom_fam)
bathroom_label_fam <- c("1", "2", "3", "4", "5", "6", "7", "10")
bathroom_fam_tab <- data.frame(bathroom_label_fam, bathroom_number_fam)

ggplot(bathroom_fam_tab) +
  geom_hline(
    aes(yintercept = y),
    data.frame(y = c(0:1) * 1000),
    color= "lightgrey") +
  geom_col(aes(
    x = reorder(str_wrap(bathroom_label_fam, 8), bathroom_number_fam),
    y = bathroom_number_fam,
    fill = bathroom_label_fam),
  position = "dodge2",
  show.legend = TRUE,
  alpha= .9) +
  geom_segment(aes(
    x = reorder(str_wrap(bathroom_label_fam, 8), bathroom_number_fam),
    y = 0,
    xend = reorder(str_wrap(bathroom_label_fam, 8), bathroom_number_fam),
    yend = 2000),
  linetype = "dashed",
  color = "gray12") +
  coord_polar() +
  labs(title = "Number of Bathrooms Preferred By Families", ylab = "Count",
       xlab="Number of Bathrooms")
```

Figure 4-67: Calculate and Visualize the Distribution of Number of Bathrooms Within Families

The family_bathroom() function is defined to calculate the number of rows for the specified number of bathroom with the condition of having family or bachelors/family as the preferred tenants. The function is called to determine the value of the variables one_bathroom_fam, two_bathroom_fam, three_bathroom_fam, four_bathroom_fam, five_bathroom_fam, six_bathroom_fam, seven_bathroom_fam and ten_bathroom_fam with passing arguments according to their respective number of bathroom value. A circular plot is then plotted with bathroom_number_fam as the x-axis and the count of each number of bathrooms as the y-axis.

Plot:

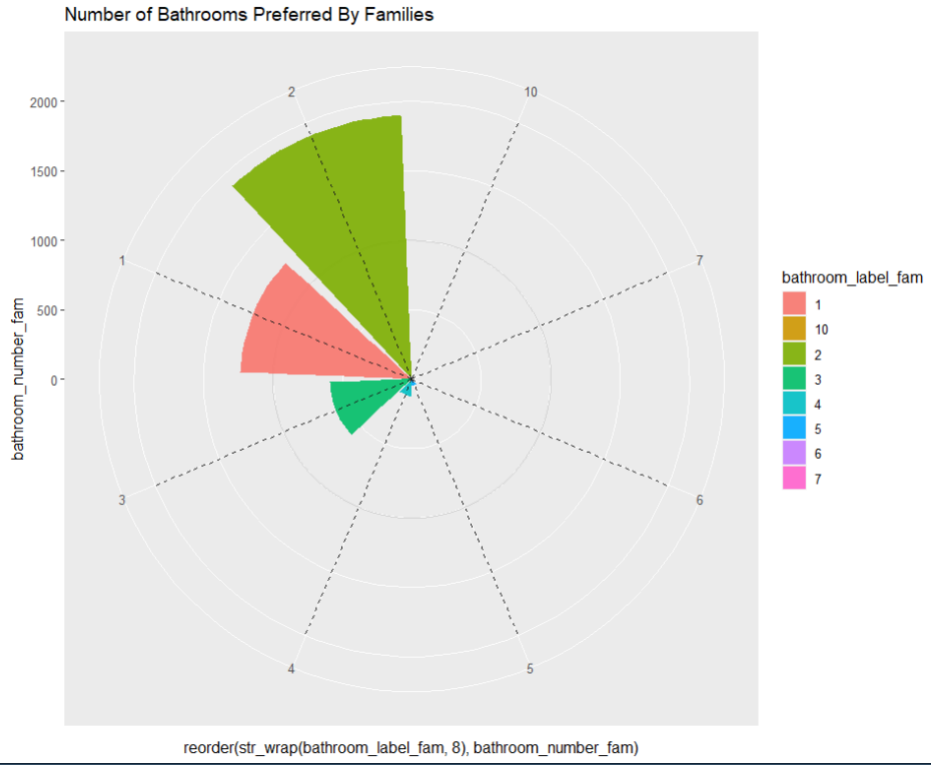

Figure 4-68: Number of Bathrooms Preferred by Families

According to the circle plot, most families prefer to rent units with 2 bathrooms, followed by 1 bathroom and 3 bathrooms. Units that consist of 4 bathrooms and above are not popular among people as the size of most families are moderate and lesser bathrooms are needed by them, people are also preventing themselves from creating spaces for waste as it is not used. 2 bathrooms are ideal to buyers as the usage of the bathrooms can be engaged in a fairer manner within the household. Having a moderate number of bathrooms can better utilize the budget used for rental as it assists in reducing the cost of rent needed.

4.4.4 Analysis 4-4: Determine the average rent price preferred by families

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The rent price of each family is extracted
Data Manipulation	The mean of the rent price of families are calculated

Source Code:

```
#Analysis 4-3: What are the average rent price preferred by families?

family_rent_price <- mean(house_data[(house_data$`Preferred Tenants`== "family"|
                                        house_data$`Preferred Tenants`== "bachelors/family"),
                                      'Rent Price'])

family_rent_price
```

Figure 4-69: Calculate the Average Rent Price for Families

The rent price of rows that fulfills the condition of having either family or bachelors/family as the preferred tenants are selected so that the mean can be calculated, which is then presented as the average rent price.

Output:

```
> family_rent_price
[1] 33480.94
```

Figure 4-70: Average Rent Price for Bachelors

According to the output, the average rent price that families prefer are 33483.94 INR, which is slightly higher than the bachelors. This could be due to families having a higher level of income as most families are established when both partners have a higher and a more stable income.

4.4.5 Analysis 4-5: What is the average unit size for families?

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The unit size of each family is extracted
Data Manipulation	The mean of the unit size of families is calculated

Source Code:

```
#Analysis 4-4: What are the average unit size preferred by families?

family_unit_size <- mean(house_data[(house_data$`Preferred Tenants` == "family" |
                                       house_data$`Preferred Tenants` == "bachelors/family"),
                                     'Unit Size'])

family_unit_size
```

Figure 4-71: Calculate the Average Unit Size for Bachelors

The unit size of rows that fulfills the condition of having either family or bachelors/family as the preferred tenants are selected so that the mean can be calculated, which is then presented as the average unit size.

Output:

```
> family_unit_size
[1] 957.3979
```

Figure 4-72: Average Unit Size for Bachelors

Based on the output, the average unit size that families prefer are 957.3979 square feet. They prefer a larger house compared to bachelors as they have more individuals staying in one house. Hence, more room is needed for guaranteed comfort for their family members.

4.4.6 Analysis 4-6: Determine the preference of families in staying at lower or upper floors

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The number of rows for each value within the floor category column for families are explored
Data Visualization	The distribution of floor ranges for families is visualized in the form of donut chart

Source Code:

```
#Analysis 4-5: Do families prefer lower or upper floors?

#Calculate the number of floors
lower_floor_fam = nrow(house_data[(house_data$`Floor Range` == "lower floors") &
                                  ((house_data$`Preferred Tenants` == "family")
                                   | (house_data$`Preferred Tenants` == "bachelors/family")),])
upper_floor_fam = nrow(house_data[house_data$`Floor Range` == "upper floors" &
                                  ((house_data$`Preferred Tenants` == "family")
                                   | (house_data$`Preferred Tenants` == "bachelors/family")),])

floor_fam = c(lower_floor_fam, upper_floor_fam)
label_fam = c("Lower Floor", "Upper Floor")

fam_floor <- data.frame(label_fam, floor_fam)

# Compute percentages
fam_floor$fraction = fam_floor$floor_fam / sum(fam_floor$floor_fam)

# Compute the cumulative percentages (top of each rectangle)
fam_floor$ymax <- cumsum(fam_floor$fraction)

# Compute the bottom of each rectangle
fam_floor$ymin <- c(0, head(fam_floor$ymax, n=-1))

# Compute label position
fam_floor$labelPosition <- (fam_floor$ymax + fam_floor$ymin) / 2

# Compute a good label
fam_floor$label <- paste0(fam_floor$label_fam, "\n value:", fam_floor$floor_fam)

#Construct Donut Plot
ggplot(fam_floor, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=label_fam)) +
  geom_rect() +
  geom_label( x=3.5, aes(y=labelPosition, label=label), size=3) +
  scale_fill_brewer(palette=4) +
  coord_polar(theta="y") +
  xlim(c(2, 4)) +
  theme_void() +
  theme(legend.position = "none") +
  ggtitle("Distribution of the Preference of Floors Among Families")
```

Figure 4-73: Calculate and Visualize the Distribution of Floor Ranges Among Families

The number of rows for each category of floor range is calculated while adhering to the conditions of having either family or bachelors/family as the preferred tenants. The results are

then combined as a vector along with the labels as another vector. Both vectors are combined to form a data frame so that a donut chart can be conveniently plotted. To start plotting the donut chart, the percentages to determine the pieces of the donut chart is computed by dividing the floor_fam column with the total sum of the column, which will be stored in a new column named as fraction. The cumulative percentages and the bottom of the rectangle are then calculated and stored as ymax and ymin respectively. The labelling is also determined by calculating its position to display the values and setting up the label value to be displayed according to the floor_fam column. Lastly, the donut plot is plotted according to the values obtained and some customizations in terms of the plot's aesthetic is done to create a more attractive plot for decision makers.

Pie Chart:

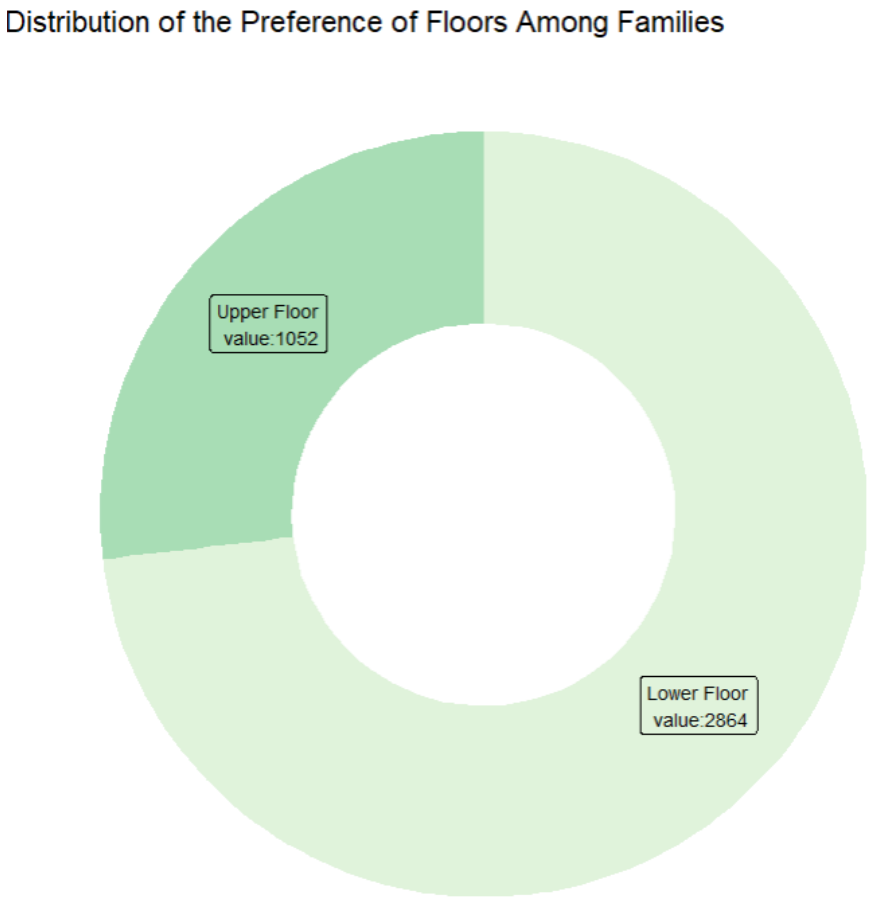

Figure 4-74: Distribution of Preferred Floor Range for Families

Based on the donut chart, it can be deduced that families prefer residing in lower floors, with having 2854 families backing with the values, which also holds the similar preference

with bachelors. Through both observations, it can also be concluded that lower floors are the most popular floor range overall. Lower floors are more popular for families as most of them consist of diverse age groups from young children to elderlies. Hence, it is more convenient for all age groups to live at lower groups as it is more accessible, and family friendly compared to higher floors where it might be taxing for certain age groups to access to. Other than that, lower floors are also easier for all age groups to evacuate when emergency occurs, making it safer than higher floors in terms of evacuation and rescue.

4.4.7 Analysis 4-7: Determine the cities that families prefer to stay at

Analysis techniques used:

Analysis Techniques	**Reason**
Data Exploration	The number of families that resides at each city is explored
Data Visualization	A pie chart is plotted to present the distribution of the number of families throughout the cities

Source Code:

```
family_city <- function(city)
{
  nrow(house_data[(house_data$`City Located` == city) &
                    ((house_data$`Preferred Tenants` == "family") |
                       (house_data$`Preferred Tenants` == "bachelors/family")),])
}

family_kolkata <- family_city("kolkata")
family_mumbai <- family_city("mumbai")
family_bangalore <- family_city("bangalore")
family_delhi <- family_city("delhi")
family_chennai <- family_city("chennai")
family_hyderabad <- family_city("hyderabad")

family_cities = c(family_kolkata, family_mumbai, family_bangalore,
                  family_delhi, family_chennai, family_hyderabad)
city_label = c("Kolkata", "Mumbai", "Bangalore", "Delhi", "Chennai", "Hyderabad")
city_number_fam = c(family_kolkata, family_mumbai, family_bangalore,
                family_delhi, family_chennai, family_hyderabad)

pie(family_cities, city_number_fam, radius = 1, main = "Distribution of Cities for Families",
    col=family_cities)
legend("topright", city_label, cex = 0.7, fill=family_cities)
```

Figure 4-75: Calculate and Visualize the Distribution of Cities for Families

The function family_city() is defined to calculate the number of rows present for each city passed as an argument, with the condition of having family or bachelors/family as the preferred tenants. The function is called to determine the values of the variables – family_kolkata, family_mumbai, family_bangalore, family_delhi, family_chennai and family_hyderabad through the city name passed as a string argument so that the number of families can be calculated. The variables are then stored in a vector, with the name of the cities as another separate vector. Via the data obtained, a pie chart is plotted with family_cities as the x-axis and city_number_fam as the labels, followed by the implementation of legends to better indicate the name of the city.

Plot:

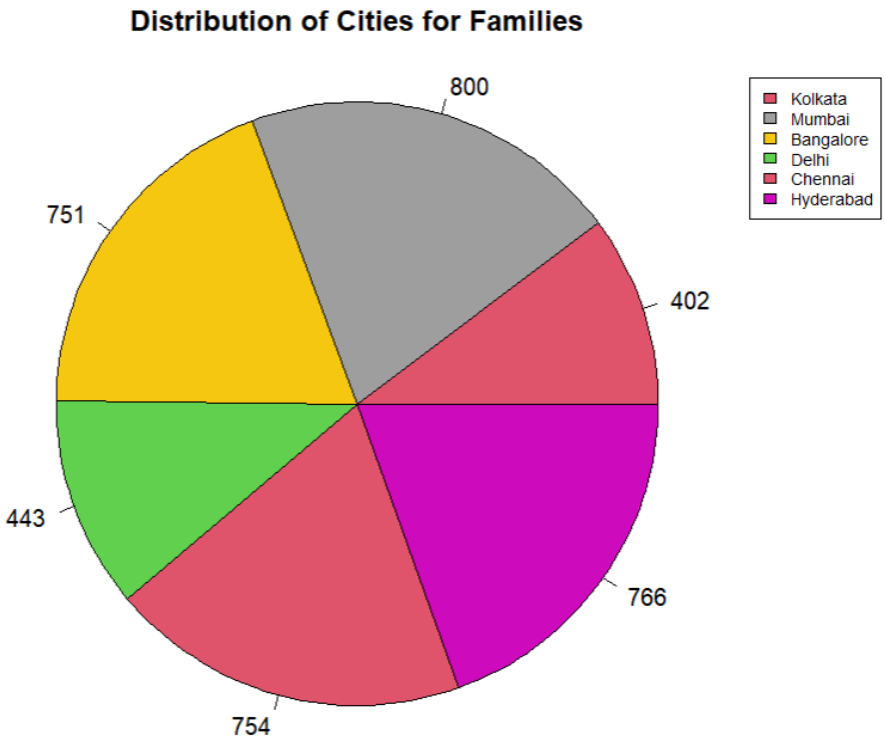

Figure 4-76: Distribution of Cities for Families

The pie chart suggests that Mumbai is the most preferred city among families to live in whereas Kolkata is the least preferred city to reside in. As families strive for a better quality of living, Mumbai seems to be the most ideal choice for them due to the city's higher level of advancement in terms of society, politics and the economy, compared with Kolkata that has faced many forms of instability and a stagnant mindset implanted within the residents of the city. Other than that, it has a higher literacy rate compared to Kolkata, which can also assist in the developing the education of the young (Placify, 2022).

4.4.8 Analysis 4-8: Determine the preference of the average total number of floors and the category of floors for families

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The number of families that prefers different types of building height is explored
Data Manipulation	The mean of the total number of floors for families is calculated
Data Visualization	A pie chart relating to the distribution of building height for bachelors are plotted

Source Code:

```
floor_mean_fam <- ceiling(mean(house_data[(house_data$`Preferred Tenants`== "family"|
                                             house_data$`Preferred Tenants`== "bachelors/family"),
                                          'Total Number of Floors']))
floor_mean_fam
```

Figure 4-77: Calculate the Mean of the Total Number of Floors for Families

The mean of the total number of floors preferred by families is calculated through rounding up the values of the mean with the ceiling() function as floors cannot exist in decimals. The mean is determined through extracting the total number of floors by taking family or bachelors/family as the preferred tenants.

```
#Stacked bar chart for total floor range
short_building_fam = nrow(house_data[(house_data$`Building Category` == "short building") &
                                       (house_data$`Preferred Tenants` == "family" |
                                          house_data$`Preferred Tenants` == "bachelors/family"),])
tall_building_fam = nrow(house_data[(house_data$`Building Category` == "tall building") &
                                      (house_data$`Preferred Tenants` == "family" |
                                         house_data$`Preferred Tenants` == "bachelors/family"),])

building_fam = c(short_building_fam, tall_building_fam)
building_label = c("Short Building", "Tall Building")
building_numbers_fam = data.frame(building_fam, building_label)

ggplot(building_numbers_fam, aes(x="", y=building_fam, fill=building_label, label=building_fam)) +
  geom_col(position = position_stack()) +
  guides(fill=guide_legend(reverse=TRUE)) +
  geom_text(hjust = 1.5, vjust=0, reverse=TRUE) +
  coord_flip() +
  ggtitle("Distribution of Building Height For Families") +
  xlab("Building Category") +
  ylab("Number of Families") +
  scale_fill_brewer(palette = 8)
```

Figure 4-78: Plotting the Stacked Bar Chart

The number of rows for short and tall buildings are calculated respectively with family and bachelors/family as preferred tenants. The variables are then stored into a vector named building_fam, followed by labels of each building category within another vector called building_label. Both vectors are combined to form a data frame to ease the visualization process. Through utilizing the data present within the data frame, a horizontal stacked bar plot is constructed with building_fam as the y-axis and label, building_label to represent the fill of the bars. The position of the columns are set to be stacked through passing the argument position_stack() and additional titles and labels were placed to properly illustrate the context of the chart.

Outcome:

```
> floor_mean_fam
[1] 7
```

Figure 4-79: Average Total Number of Floors Families Prefer to Rent

Similar to bachelors, the average total number of floors families prefer to rent is 7 floors. Hence, it can be concluded that the preference of the building height does not vary between different marital status rather through personal ideals and personalities.

Plot:

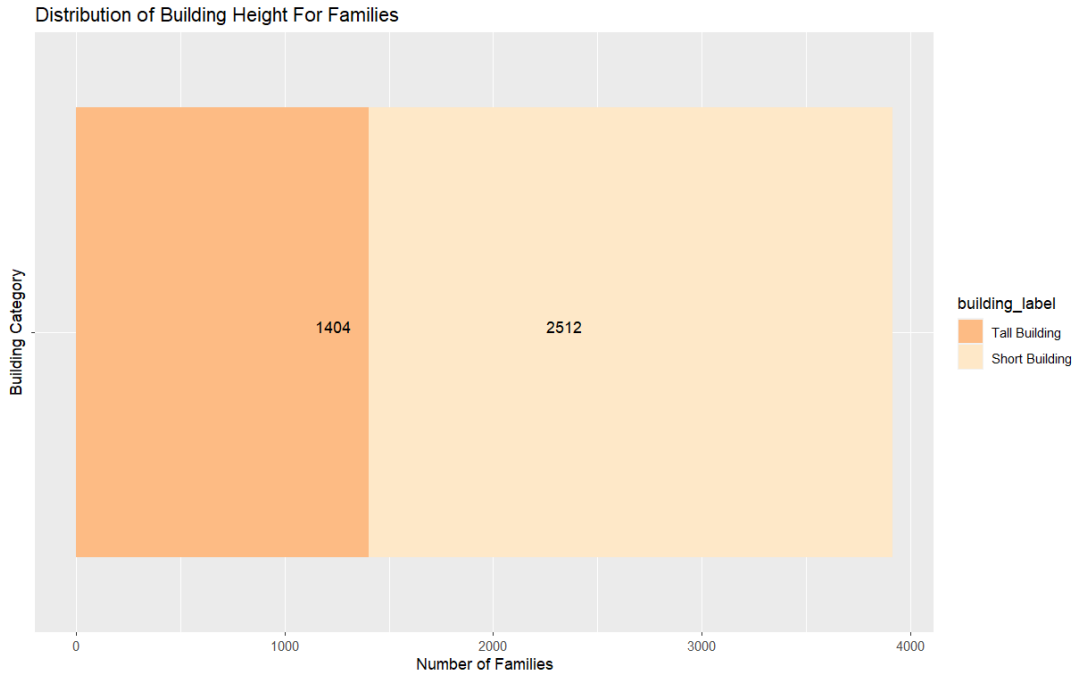

Figure 4-80: Distribution of Building Height for Families

The horizontal stacked bar chart illustrates that 2512 families prefer to stay at short buildings whereas 1404 families prefer to stay at tall buildings, which has also indicated that most family's preference is to stay at short buildings. Short buildings increase the chance of propinquity and the liveliness of the building as families do prefer an environment that is more warm and friendly. Hence, also improving the living conditions of the family through closer physical or psychological proximity between people (Bloomingrock, n.d.). Professionals have also suggested that short buildings increase the likelihood of elderlies and children to exercise extra efforts to participate in outdoor activities to reduce isolation and alienation from society.

4.4.9 Analysis 4-9: Determine the preference of the type of area for families

Analysis techniques used:

Analysis Technique	Reason
Data Exploration	The number of rows of families for each area type is explored
Data Visualization	A bar chart is plotted according to the number of families for each area type

Source Code:

```
family_area <- function(area)
{
  nrow(house_data[(house_data$`Type of Area` == area) &
                    (house_data$`Preferred Tenants` == "family" |
                       house_data$`Preferred Tenants` == "bachelors/family"),])
}

super_area_fam = family_area("super")
carpet_area_fam = family_area("carpet")
built_area_fam = family_area("built")

#Plot bar chart for type of area
area_type_fam <- c(super_area_fam,carpet_area_fam,built_area_fam)
area_label <- c("Super Area", "Carpet Area", "Built Area")
fam_area <- data.frame(area_label, area_type_fam)

ggdotchart(fam_area, x = "area_label", y = "area_type_fam",
           color="area_label",
           palette = c("#00AFBB", "#E7B800", "#FC4E07"),
           sorting = "descending",
           rotate = TRUE,
           y.text.col = TRUE,
           dot.size = 15,
           label = area_type_fam,
           font.label = list(color = "white", size = 9,
                             vjust = 0.5),
           ggtheme = theme_pubr()
  )+
  theme_cleveland() +
  ggtitle("Distribution of The Area Types for Families") +
  ylab("Number of Families")
```

Figure 4-81: Calculate and Visualize the Area Types for Families

The function family_area() is defined to calculate the number of rows present for each area according to the condition of having family or bachelors/family as preferred tenants. The values retrieved are combined into a data frame so that the Cleveland's dot plot can be plotted. The dot chart is plotted with the ggdotchart() function where the grouping is done through the

use of colors and the data is sorted in a descending order horizontally, labels are added as well to enable decision makers to accurately gage the total number of families.

Plot:

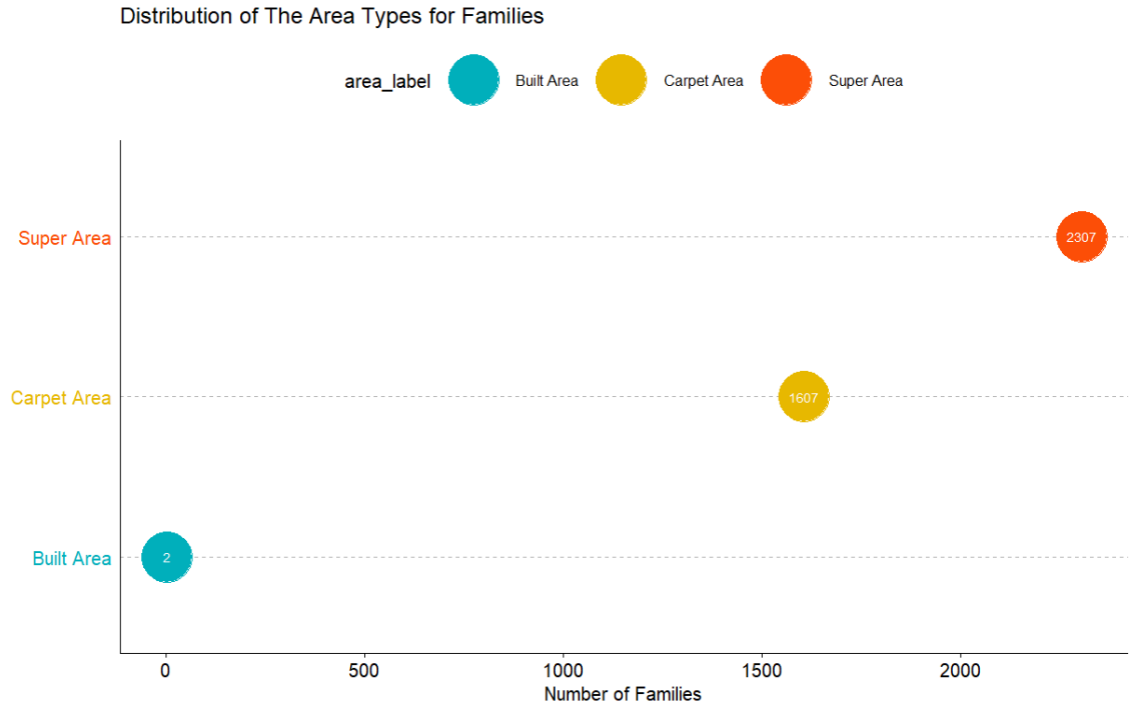

Figure 4-82: Distribution of Area Types for Families

Based on the dot plot, super area is the most preferred types of area measurement among all three. Families do prefer such type of measurement as it only consists of areas that are usable, which can help in determining a more reasonable price to be paid for the unit. The price determined by the area measurement also includes common construction such as lifts and stairs. The addition to that is it also covers amenities like the gym, pool, clubhouse and garden. Families tend to take the presence of facilities into account to ensure that everyone in the family lives a balanced and healthy lifestyle. Other than that, the piping and air ventilation are included into the pricing, which might also reduce the cost of maintenance in the future as it can be well maintained with the cost funded and prevent hazards.

4.4.10 Analysis 4-10: Determine the preference of the type of furnishings for families

Analysis techniques used:

Analysis Technique	Reason
Data Exploration	The number of rows of families for each furnishing type is explored
Data Visualization	A lollipop chart is plotted according to the number of families for each furnishing type

Source Code:

```
family_furnishing <- function(furnishing)
{
  nrow(house_data[(house_data$Furnishings == furnishing) &
                    (house_data$`Preferred Tenants`== "family" |
                       house_data$`Preferred Tenants` == "bachelors/family"),])
}

unfurnished_fam = family_furnishing("unfurnished")
semi_furnished_fam = family_furnishing("semi-furnished")
furnished_fam = family_furnishing("furnished")

#Plot Lollipop chart for furnishings
furnishing_type_fam <- c(unfurnished_fam, semi_furnished_fam, furnished_fam)
furnishing_label <- c("Unfurnished", "Semi-furnished", "Furnished")
furnish_fam_tab <- data.frame(furnishing_label, furnishing_type_fam)

ggdotchart(furnish_fam_tab, x = "furnishing_label", y = "furnishing_type_fam",
           color = "furnishing_label",
           palette = c("#00AFBB", "#E7B800", "#FC4E07"),
           sorting = "descending",
           add = "segments",
           rotate = TRUE,
           group = "furnishing_label",
           dot.size = 10,
           label = round(furnish_fam_tab$furnishing_type_fam),
           font.label = list(color = "white", size = 9,
                             vjust = 0.5),
           ggtheme = theme_pubr()
) + ggtitle("Distribution of Furnishing Types for Families") +
  xlab("Number of Family") +
  ylab("Furnishing Type")
```

Figure 4-83: Calculate and Visualize Distribution of Furnishings for Families

The family_furnishing() function is responsible in calculating the number of rows present for each type of furnishing for families. The values obtained through the function according to the strings passed is stored inside a vector. A data frame is created with the furnishing_label and furnishing_type_fam vectors. A lollipop chart is then plotted with the ggplotchart() function, which will be explained further in the Extra Feature 11: ggplotchart().

Plot:

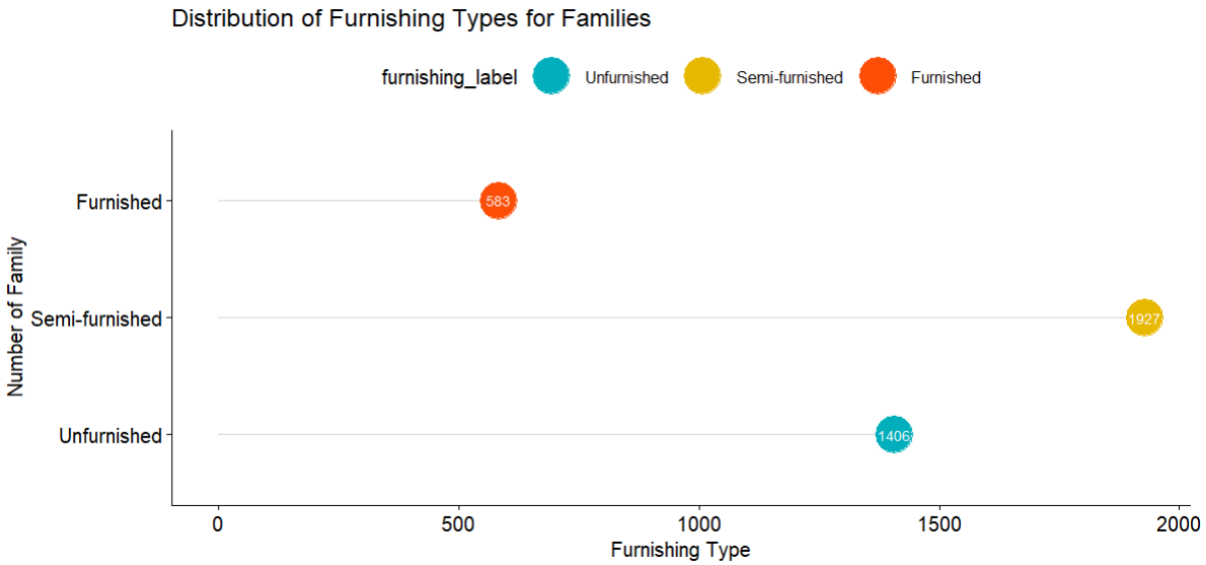

Figure 4-84: Distribution of Furnishing Types for Families

According to the lollipop plot, semi-furnished units are seen to be the most preferred type of furnishing within families, which is also like the preferences of bachelors. Families prefer such units as it is easier to adapt to all age groups if the necessities and utilities are present within the unit. There will be spaces to spare to move in furniture and appliances according to their demands and needs to live normally daily. Other than that, age group-oriented items such as baby chairs and walkers can be conveniently place within the house without having to consider the fixed items.

4.4.11 Analysis 4-11: Determine the preference of contact person for families

Analysis techniques used:

Analysis Technique	Reason
Data Exploration	The number of rows of families for each contact person is explored
Data Visualization	A bar chart is plotted according to the number of families for each contact person

Source Code:

```
family_contact <- function(person)
{
  nrow(house_data[(house_data$`Contact Person` == person)
                  & (house_data$`Preferred Tenants`== "family" |
                       house_data$`Preferred Tenants` == "bachelors/family"),])
}

owner_fam = family_contact("owner")
agent_fam = family_contact("agent")
builder_fam = family_contact("builder")

#Plot bar chart for contact person
contact_person_fam <- c(owner_fam, agent_fam, builder_fam)
contact_label <- c("Owner", "Agent", "Builder")

contact_fam <- barplot(contact_person_fam, main = "Distribution of Contact Person for Families",
                  xlab = "Person of Contact", ylab = "Number of Families", names.arg = contact_label,
                  col = contact_person_fam, ylim = c(0,3000))
text(contact_fam, 0, contact_person_fam, cex = 1, pos = 3)
```

Figure 4-85: Calculate and Visualize the Distribution of Contact Person for Families

The family_contact() function is defined to calculate the number of rows present for family or bachelors/family based on the contact person selected. The function is called for the owner, agent and builder variables along with passing the arguments – owner, agent and builder respectively in the form of strings. Then, a bar chart is plotted according to the outcome of the variables, which is stored in a vector.

Plot:

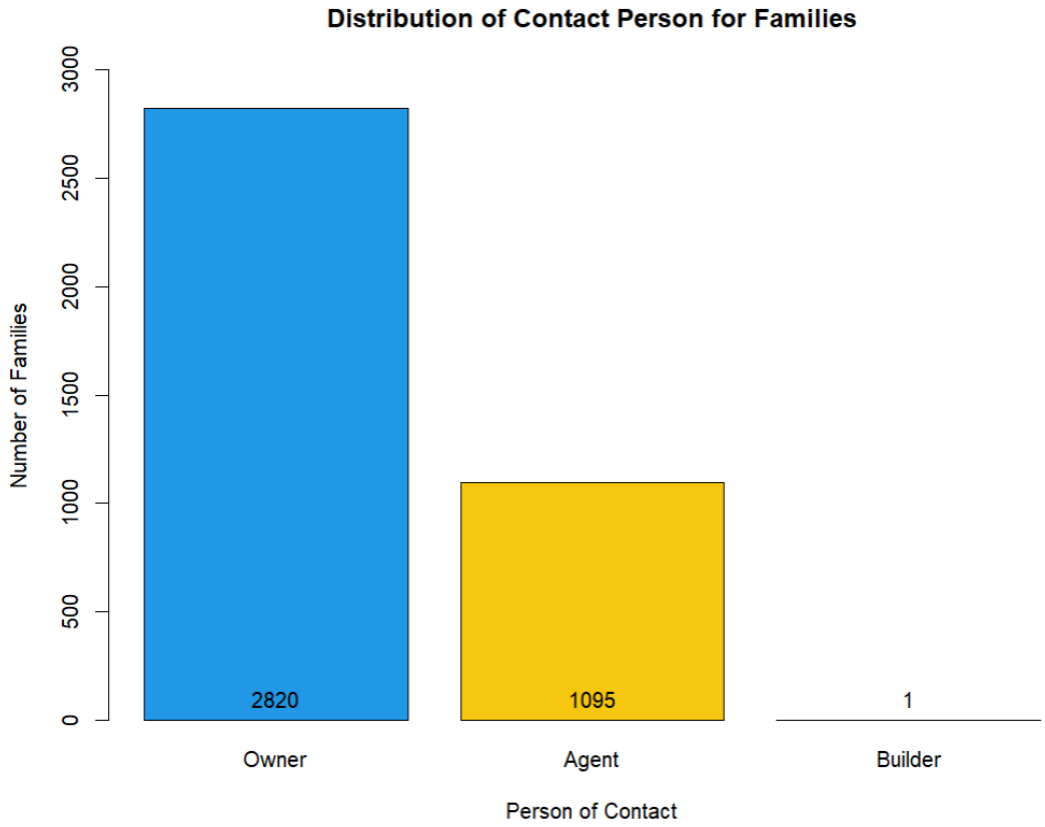

Figure 4-86: Distribution of Contact Person for Families

Similar to bachelors, families prefer to contact the owner of the house during the inquiry phase, followed by agents and builders as they believe that they can obtain more genuine and reliable information from them rather than getting the probability of receiving inaccurate information from external parties such as agent and builder, which might affect their decision making while renting the unit.

4.4.12 Conclusion

By analyzing the preference of families, it can be concluded that families consider many factors in selecting their desired house for rental. The most popular preference of house is a unit with 2 BHK which consists of the following characteristics:

1. 2 bathrooms
2. Has an average rent price of 33480.94 INR
3. A unit size of 957.4 square feet
4. Located at the lower floors
5. Shorter buildings
6. Located in Mumbai
7. Super areas
8. Semi-furnished units
9. Having the owner as the main contact person

The characteristics stated has also determined that families prefer units that encourages them to practice healthier lifestyles, practical, slightly budget, not oversized, comfortable, located in areas with a moderate to higher cost of living as the quality of living matters to them, has a moderate level of customization in terms of furnishings and has a contact person that is more convenient to enquire so that more reliable information can be conveyed.

On the other hand, families put the least preferences on units with 6 BHK and above and consist of the following characteristics:

1. Has 4 or more bathrooms
2. Unit sizes more than 1200 square feet
3. Located at upper floors
4. Taller buildings
5. Located in Kolkata
6. Built areas
7. Furnished units
8. Having the builder as the main contact person

Compared to the characteristics obtained from the bachelors earlier, it has a high similarity. Therefore, it can be deduced that most of the people do not enjoy having the characteristics as stated for their houses. Families do not prefer units that consist of too big of the space to live in, too many bathrooms that will be unused by family members, too high for elderlies and children to access and having a lower level of customization in terms of furnishing. Kolkata is not preferred by families as it has lesser varieties of properties, which might not be ideal for house selection.

4.5 Question 5: How the locality and city affect the number of floors of houses?

4.5.1 Parse the locality to two separate columns

Source Code:

```
#Parse the locality to two columns
house_data[c('Area', 'Area 2')] <- str_split_fixed(house_data$Locality,', ', 2)
house_data$`Area 2`=NULL
unique(house_data$Area)
```

Figure 4-87: Separate the Locality to Two Columns

The localities are split to 2 columns – Area and Area 2, it is done by using the str_split_fixed() function with "," as the separator. After that, the Area 2 column is dropped as it is not needed further for analysis due to the vast number of null values. The Area column is then explored in a deeper level through utilizing the unique() method.

Outcome:

Area
bandel
phool bagan
salt lake city sector 2
dumdum park
south dum dum
thakurpukur
malancha
malancha
palm avenue kolkata
natunhat
action area 1
keshtopur
tarulia
dum dum metro
paschim barisha
new town action area 1
barasat
behala
behala chowrasta
behala
santoshpur
garia station
garia station
joka

Figure 4-88: Area Column

The Area column consists of the first part present in the locality before the delimiter ",", which covers a diverse list of localities from different cities to ease and short the visualizations present.

4.5.2 Analysis 5-1: Distribution of the height of a building according to city

Analysis techniques used:

Analysis Techniques	Reason
Data Visualization	The height of the building is visualized as bar charts according to cities

Source Code:

```
ggplot(house_data, aes(x =`Building Category`, fill=`Building Category`)) +
  geom_bar() +
  facet_wrap(~`City Located`) +
  labs(title = "Distributions of Building Categories According to Cities")
```

Figure 4-89: Visualize Bar Chart According to Cities

To visualize the bar chart, the ggplot() function, the Building Category is placed as the x-axis and the fill. By implementing geom_bar(), the graph will appear in the form of bar plots along with facet_wrap() for the City Located to create separate plots for each city. The plot is then given the title – Distributions of Building Categories According to Cities.

Plot:

Figure 4-90: Distributions of Building Categories According to Cities

According to the bar charts, most of the cities tend to have more short buildings compared to tall buildings, except for Mumbai who has a significantly larger number of tall buildings. Chennai has the largest number of short buildings whereas Mumbai has the smallest number of such buildings. This might be due to Mumbai being a rapidly developing city in the country. Hence, more tall buildings and skyscrapers were built. As for Chennai, it is not encouraged to construct taller buildings as the Weather Radar installed by the Indian government within Chennai city is lower in terms of height.

4.5.3 Analysis 5-2: Mean of total height of buildings in each city

Analysis techniques used:

Analysis Techniques	Reason
Data Manipulation	To calculate the mean of the total height of buildings through grouping by the cities and sort values according to the mean
Data Visualization	A lollipop chart is plotted to visualize the values and the values above and below the mean

Source Code:

```
floor_means <- house_data %>%
  group_by(`City Located`) %>%
  summarise(mean_floor=ceiling(mean(`Total Number of Floors`)))
floor_means

city_floor <- floor_means %>%
              arrange(mean_floor) %>%
              mutate(Avg = mean(mean_floor, na.rm = TRUE),
                     Above = ifelse(mean_floor - Avg > 0, TRUE, FALSE),
                     city = factor(`City Located`, levels = .$`City Located`))
city_floor

ggplot(city_floor, aes(y =`City Located`, x=mean_floor, color=Above, label=mean_floor)) +
  geom_segment(aes(x=Avg, y=`City Located`, xend = mean_floor, yend =`City Located`), color = "grey50") +
  geom_point() +
  geom_text(nudge_x = 1.5) +
  labs(title = "Mean of Total Height of Buildings In Each City") +
  xlab("Mean of Total Floors")
```

Figure 4-91: Manipulate and Visualize Mean of the Total Height of Buildings in Each City

The means of the total number of floors based on each city is calculated and stored in a data frame named floor_means through the use of group_by() and summarise() function present in the dplyr package. The mean calculated is rounded up to the nearest whole number as floors cannot exist in the form of decimals. Then, the floor_means data frame is taken so that the values in the mean_floor column can be arranged. Several new columns, which includes Avg, Above and City is derived to assist in visualizing the values above and below the mean of the total number of floors according to each city, with the help of the mutate() function. Avg is obtained by calculating the mean of the mean_floor column, while the Above column is formulated by subtracting the Avg variable with the mean_floor. Values that are more than 0 are categorized as true, which means that it is above the mean value, whereas false indicates that the value is below the mean. The city column categorizes the value according to the City Located column. Through the new data frame obtained, city_floor, the lollipop plot is plotted

by initiating the City Located as the y-axis, mean_floor as the x-axis, above as the color indicators to better differentiate the difference between both categories and mean_floor as the label. The geom_segment function is to illustrate the lines present in the plot according to the values away from the mean point and geom_point is used to draw the dots at the end of the lollipop line.

Plot:

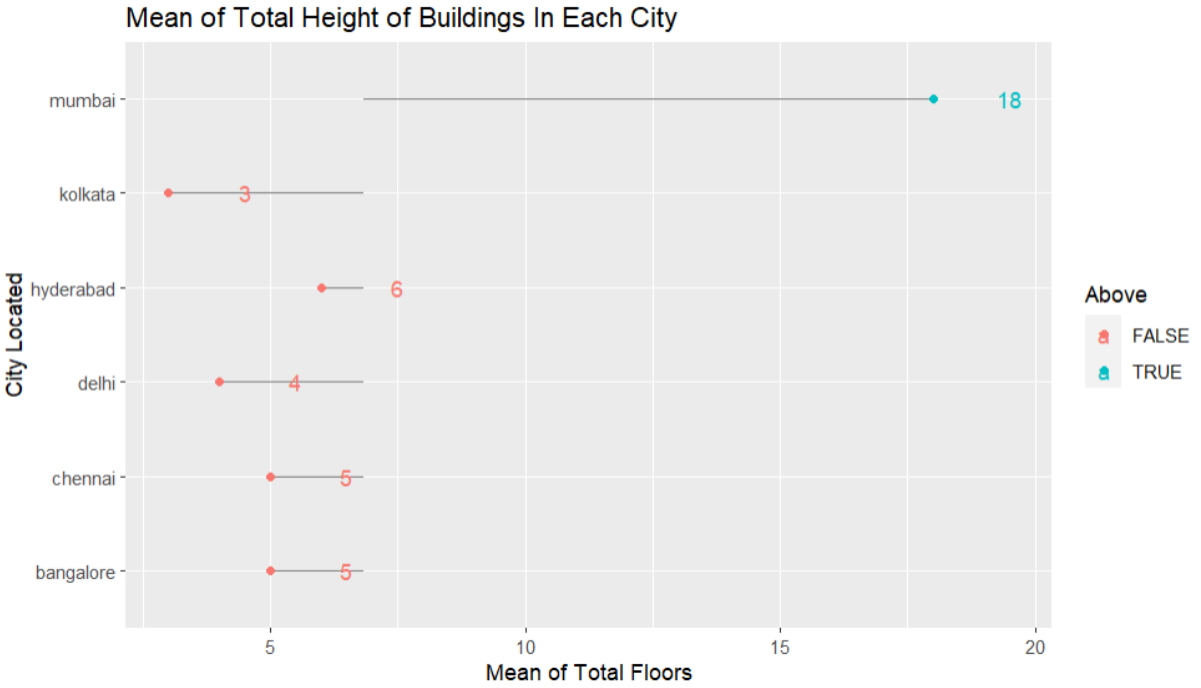

Figure 4-92: Mean of Total Height of Buildings in Each City

According to the lollipop chart, Mumbai is the only city with average height of buildings more than the mean, which can also mean that the city consists of many high-rise buildings as it could be due to the increasing population and demand of evolution within the city. On the other hand, the mean height of buildings in Kolkata is the lowest, meaning that the city mainly consists of short buildings. The structure of the buildings could be affected due to the government policy, cost, societal knowledge, environmental and geographical factors. Hyderabad's mean total of building height is close to the mean, which can also be categorized as having more moderate height buildings within the city compared to the other cities.

4.5.4 Analysis 5-3: Determine the top localities

Analysis techniques used:

Analysis Techniques	Reason
Data Manipulation	The frequency of each locality is counted and the top 10 is extracted and added into a new data frame
Data Visualization	A bar plot is visualized to determine the top localities in house rentals

Source Code:

```
#Extract the top 10 locality
locality_freq <- data.frame(house_data %>% count(Area, sort=TRUE))
top_ten <- head(locality_freq, 10)

ggbarplot(top_ten, x = "Area", y="n",
          fill = "Area",
          color = "white",              # Set bar border colors to white
          palette = "jco",              # jco journal color palette
          sort.val = "desc",            # Sort the value in descending order
          sort.by.groups = FALSE,       # Don't sort inside each group
          x.text.angle = 90,
          label = TRUE) +
  ggtitle("Distibution of Properties in Top 10 Localities") +
  ylim(0,40)
```

Figure 4-93: Extract and Visualize Top Localities

The frequency of each area is counted via the count() function and is sorted in descending order, the data is converted to a data frame to ease visualization. To prevent the IDE from crashing due to the overwhelming number of data to visualize, the top 10 localities are selected to be placed in the top_ten data frame using the head() function while passing locality_freq as the data and 10 as the desired number of rows to be selected. The ggbarplot() function is used to illustrate a bar chart of the distribution of properties in the top 10 localities. The top_ten data frame is passed as the data, Area as the x-axis and fill, n as the y-axis with some color customization while sorting the values in the descending order as the top values can be seen easily. However, the bars are not sorted within each groups.

Plot:

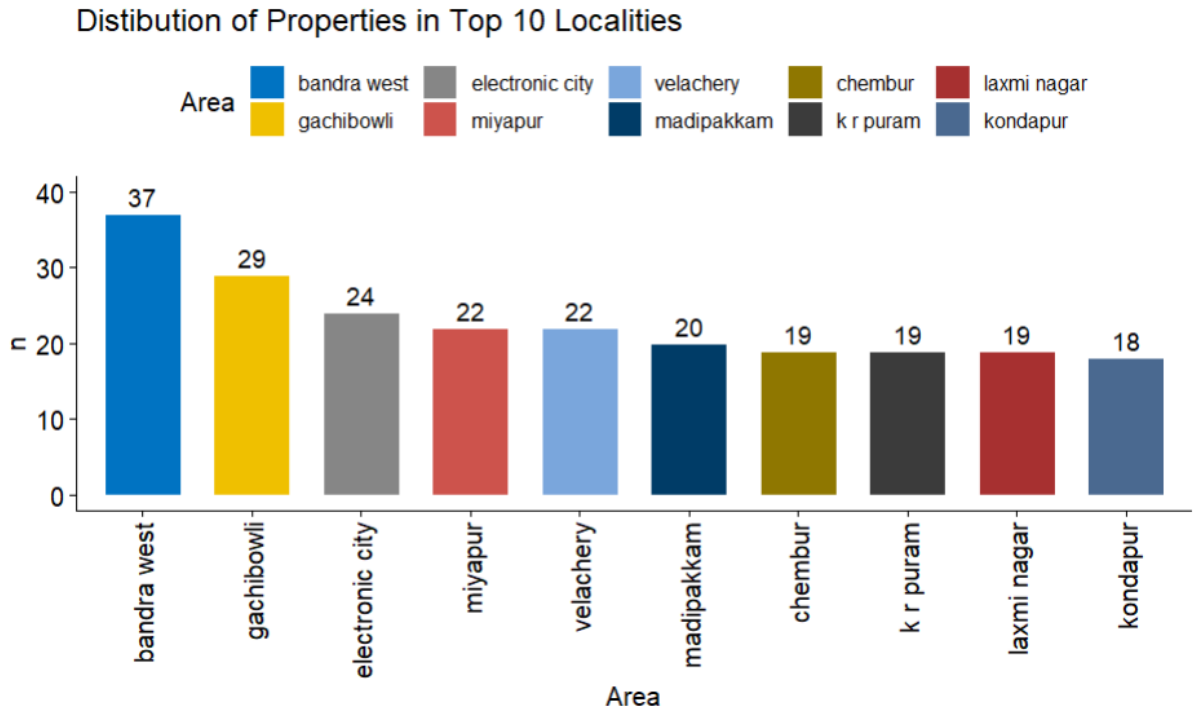

Figure 4-94: Top 10 Localities

The top 10 localities consist of the areas – Brandra West, Gachibowli, Electronic City, Miyapur, Velachery, Madipakkam, Chembur, K R Puram, Laxmi Nagar and Kondapur. Bandra West located in Mumbai holds the highest record of properties, Gachibowli located in Hyderabad as second highest and Electronic City located in Bangalore as the third highest. The number of properties available within the localities varies according to the popularity of the area, the shops and facilities available and population present to accommodate their needs.

4.5.5 Analysis 5-4: The mean total number of floors according to the top 5 localities

Analysis techniques used:

Analysis Techniques	Reason
Data Manipulation	- The mean of the total number of floors for the top 5 localities are calculated - Creating a new data frame with the average, value category and locality as the values within each column
Data Visualization	A lollipop plot is illustrated to represent the mean calculated for the top 5 localities and its distance from the mean total number of floors

Source Code:

```
locality_floor_mean <- house_data %>%
  group_by(Locality) %>%
  summarise(locality_mean=ceiling(mean(`Total Number of Floors`)))
locality_floor_mean

localities <- c('bandra west', 'gachibowli', 'electronic city', 'miyapur, nh 9', 'velachery')

top_five_mean <- locality_floor_mean[locality_floor_mean$Locality %in% localities,]

locality_floor <- top_five_mean %>%
  arrange(locality_mean) %>%
  mutate(average = mean(locality_mean, na.rm = TRUE),
         above = ifelse(locality_mean - average > 0, TRUE, FALSE),
         locality = factor(Locality, levels = .$Locality))
locality_floor

ggplot(locality_floor, aes(y =Locality, x=locality_mean, color=above, label=locality_mean)) +
  geom_segment(aes(x=average, y=Locality, xend = locality_mean, yend =Locality), color = "grey50") +
  geom_point() +
  geom_text(nudge_x = 1.5) +
  labs(title = "Mean of Total Height of Buildings In The Top 5 Localities To Stay At") +
  xlab("Mean of Total Number of Floors")
```

Figure 4-95: Manipulate and Visualize the Mean of Building Height According to Localities

To construct the locality_floor_mean data frame, the columns from the house_data data frame is taken. The mean of the total number of floors are calculated with the summarise() function and is grouped according to the localities with the group_by() function. The names of the top 5 localities are then stored into a vector based on the observations made in Analysis 5-3. The top_five_mean data frame is made by extracting rows with the localities present within the localities vector using %in%. Then, the locality floor data frame is constructed through

arranging the locality_mean column in the top_five_mean data frame and aggregated three new columns – average, above, locality with the mutate() function. Lastly, the lollipop plot is charted with the ggplot() function, where the locality_floor data frame is used, with Locality represented as the y-axis, locality_mean as the x-axis and label, above as the color.

Plot:

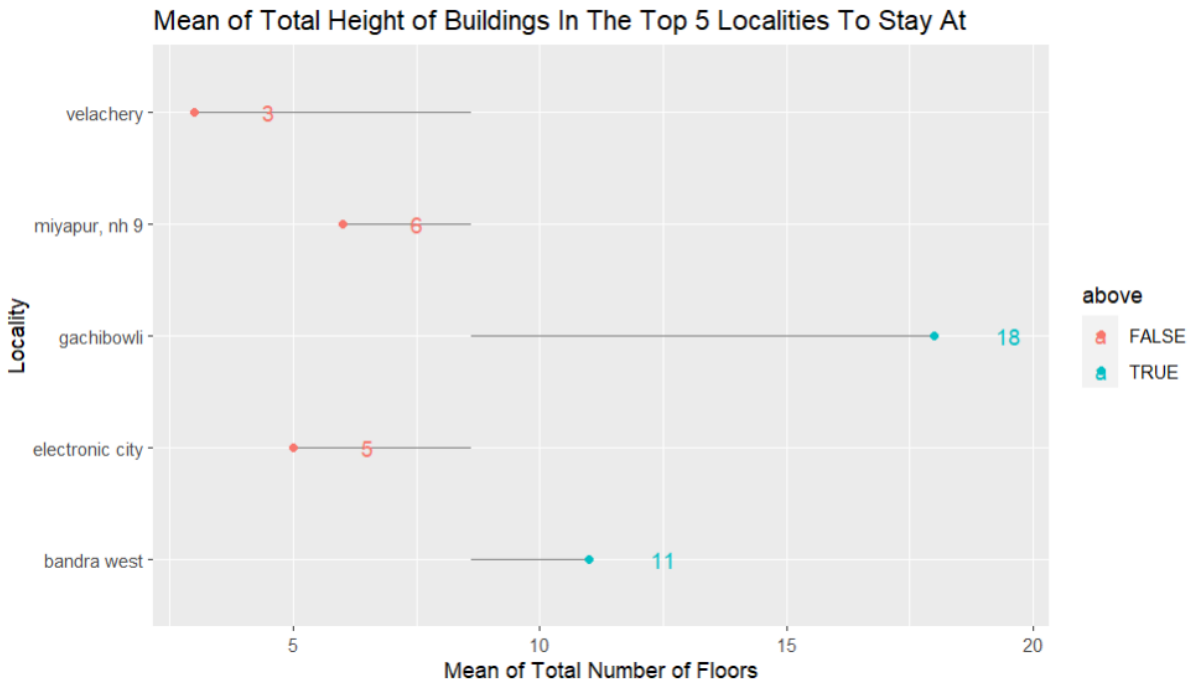

Figure 4-96: Mean of Total Height of Buildings in The Top 5 Localities

The lollipop plot has stated that Gachibowli has the highest mean of total number of floors, followed by Bandra West where both localities are listed above the mean. Through the results obtained, it can be deduced that both cities have a higher likelihood for residences to reside in high-rise buildings and skyscrapers, along with having better facilities to facilitate the quality of life for residents due to their advancement in terms of architectural and structural engineering. On the other hand, the mean of the total height of building for Velachery, Miyapur and Electronic City are more towards the lower side as it is below the stated mean, but with Velachery as the lowest among all. This could also be due to geographical or economy constraints.

4.5.6 Analysis 5-5: Determining the geographical location of the cities

Analysis techniques used:

Analysis Techniques	**Reason**
Data Manipulation	- Columns relating to the latitude and longitude of the cities are added into the data frame - The distinct values of the cities are extracted and stored into a data frame - Columns are extracted from the data frame - The column names of the data frame are renamed
Data Visualization	The shape of the India map is plotted with the cities present on the map

Source Code:

```
house_data <- house_data %>%
  mutate('Latitude' = case_when(
    `City Located`== 'kolkata' ~ 22.5726,
    `City Located`== 'mumbai' ~ 19.0760,
    `City Located`== 'bangalore' ~ 12.9716,
    `City Located`== 'delhi' ~ 28.6448,
    `City Located`== 'chennai' ~ 13.0827,
    `City Located`== 'hyderabad' ~ 17.3850
  ))

house_data <- house_data %>%
  mutate('Longitude' = case_when(
    `City Located`== 'kolkata' ~ 88.3639,
    `City Located`== 'mumbai' ~ 72.8777,
    `City Located`== 'bangalore' ~ 77.5946,
    `City Located`== 'delhi' ~ 77.2167,
    `City Located`== 'chennai' ~ 80.2707,
    `City Located`== 'hyderabad' ~ 78.4867
  ))

india <- map_data("world", region = "india")
india_plot <- ggplot() +
  geom_polygon(data=india, aes(x=long, y=lat, group=group), fill="lightgreen") +
  coord_fixed(1.3)

cords <- house_data %>% distinct(`City Located`, .keep_all = TRUE)
cords <- select(cords,`City Located`, Longitude, Latitude)
names(cords) <- c('City', 'Longitude', 'Latitude')
cords

india_plot +
  geom_point(data=cords, aes(x=Longitude, y=Latitude, color=City), size = 3) +
  geom_text(data=cords, aes(x=Longitude, y=Latitude, label=paste(" ",
                                                                 as.character(City), sep = "")),
            hjust = 0.5, vjust=1.5) +
  labs(title="Geographical Locations of Cities")
```

Figure 4-97: Transforming and Plotting Map

To plot the map of India with the cities present, the latitude and longitude has to be determined. The columns are made within the house_data data frame using the mutate() function while integrating the case_when() function so that the values can be assigned according to the cities selected for both columns. The map of India is then imported with the help of map_data(). To construct the shape of India, the ggplot() function is used and the India map is passed as the data, long as x-axis, lat as y-axis and group as group within the geom_polygon() function with the help of the coord_fixed() function to adjust the aspect ratio of the plot. The distinct values of the City Located column are extracted and stored in the cords variable, then the columns – City Located, Longitude and Latitude are selected to be stored in the same variable as a data frame. Lastly, the points that represents the location of each city within the map is plotted with the city's label through geom_point() and geom_text() by taking the data from the latitude and longitude columns in the cords data frame.

Map Plot:

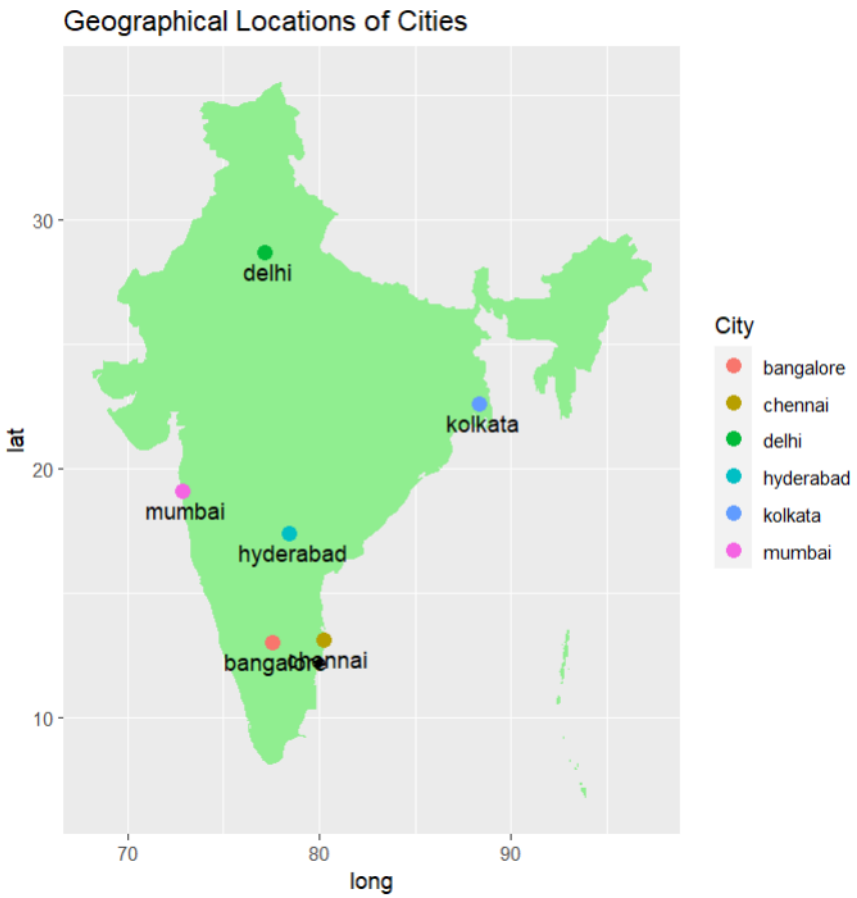

Figure 4-98: Geographical Locations of Cities

According to the map, it can be determined that cities such as Mumbai, Kolkata and Chennai are coastal areas as they are located at the sides of the map of India. On the other hand, cities like Delhi, Hyderabad and Bangalore are inland areas as the cities are covered with land and accessing the coast is not that convenient compared to coastal areas.

Determine Map Border Coordinates

Extra Feature 10: map_data()

4.5.7 Analysis 5-6: Distribution of properties according to the geographical structure

Analysis techniques used:

Analysis Techniques	Reason
Data Exploration	The count of properties for each geographical location is explored
Data Manipulation	The Geographical Location column is added according to the data obtained from Analysis 5-7
Data Visualization	A pie chart is plotted to view the distribution of properties according to geographical location

Source Code:

```
house_data <- house_data %>%
  mutate('Geographical Location' = case_when(
    `City Located`== 'kolkata' ~ 'coastal',
    `City Located`== 'mumbai' ~ 'coastal',
    `City Located`== 'bangalore' ~ 'inland',
    `City Located`== 'delhi' ~ 'inland',
    `City Located`== 'chennai' ~ 'coastal',
    `City Located`== 'hyderabad' ~ 'inland'
  ))

geographical_data <- house_data %>% count(`Geographical Location`)

ggplot(geographical_data, aes(x="", y=n,
                      fill = `Geographical Location`)) +
  geom_col() +
  geom_text(aes(label=n),
            position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  scale_fill_brewer(palette = 9) +
  labs(title="Distribution of Properties According to Geographical Location")
```

Figure 4-99: Explore, Manipulate and Visualize Geographical Structure

The mutate() function is used with the case_when() function to create a new column within the house_data data frame by assigning the categories of the geographical location to the cities present. The count() function is called to count the total number of properties for each geographical location and is stored in the geographical_data data frame. A pie chart is then plotted with the n variable with geographical location as the fill. Labels are also included with

the help of geom_text() to ease readability for the chart and other custom aesthetics settings such as scale_fill_brewer() and labs() are used to enhance the design.

Plot:

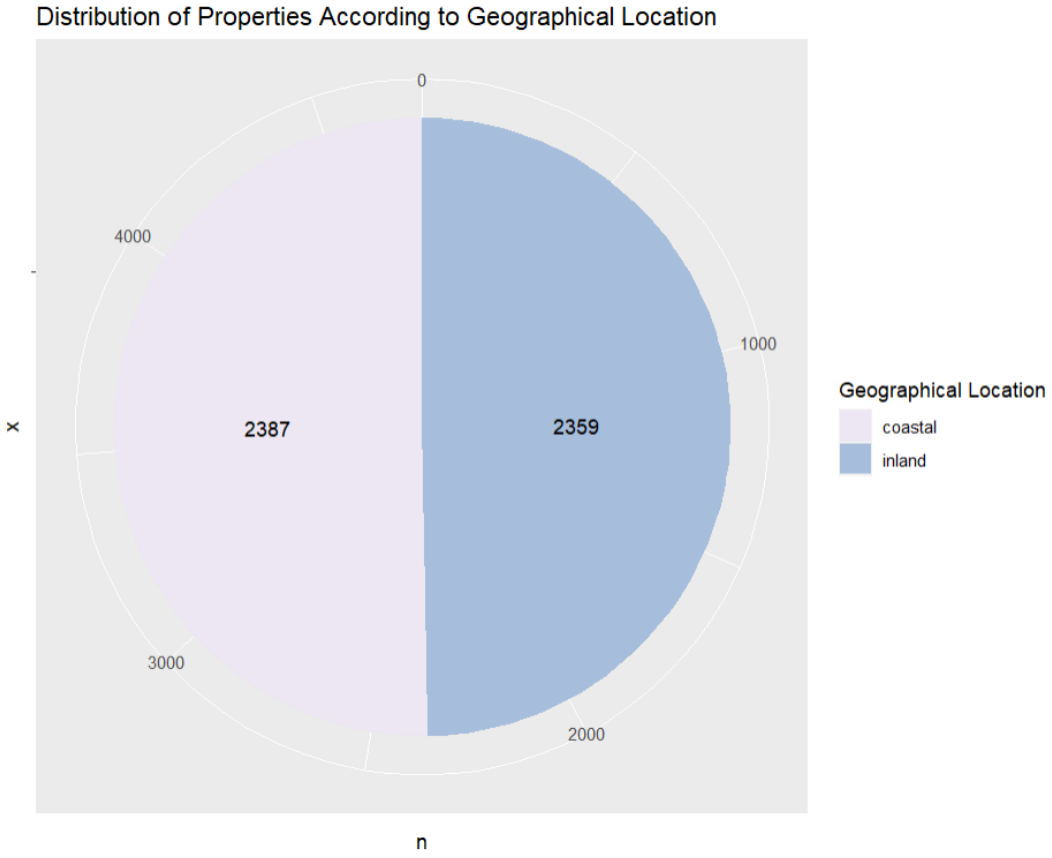

Figure 4-100: Distribution of Properties According to Geographical Location

The pie chart has illustrated a nearly equal distribution of properties for both geographical locations. This could be due to residences having equal preferences on staying on either coastal or inland areas and the geography of both type of lands are friendly for constructing new buildings. Hence, no drastic effects are brought while building on both types of geographical location.

4.5.8 Analysis 5-7: Distribution of the average total number of floors according to geographical characteristics

Analysis techniques used:

Analysis Techniques	Reason
Data Manipulation	The mean of the total number of floors for each geographical location are calculated
Data Visualization	A bar plot is illustrated to represent the mean calculated for the geographical locations

Source Code:

```
geographical_floor <- house_data %>%
  group_by(`Geographical Location`) %>%
  summarise(geo_mean=ceiling(mean(`Total Number of Floors`)))
geographical_floor

ggplot(geographical_floor, aes(x=`Geographical Location`, y=geo_mean,
                               fill=`Geographical Location`, label=geo_mean))+
  geom_bar(stat="identity") +
  geom_text() +
  scale_fill_manual(values = c("#FF007F", "#7F00FF")) +
  labs(title="Average Total Number of Floors According to Geographical Characteristics")
```

Figure 4-101: Manipulate and Visualize the Average Total Number of Floors According to Geographical Characteristics

To construct the geographical_floor data frame, the columns from the house_data data frame is taken. The mean of the total number of floors are calculated with the summarise() function and is grouped according to the geographical location with the group_by() function. A bar chart is plotted with the ggplot() function using the geographical_floor data frame by assigning Geographical Location as the x-axis and fill, geo_mean as the y-axis and label. The bar is set to identity as stats so that the heights of the bars represent the values. Other aesthetic features such as geom_text(), scale_fill_manual() and labs() are implemented to improve the overall outlook of the plot.

Plot:

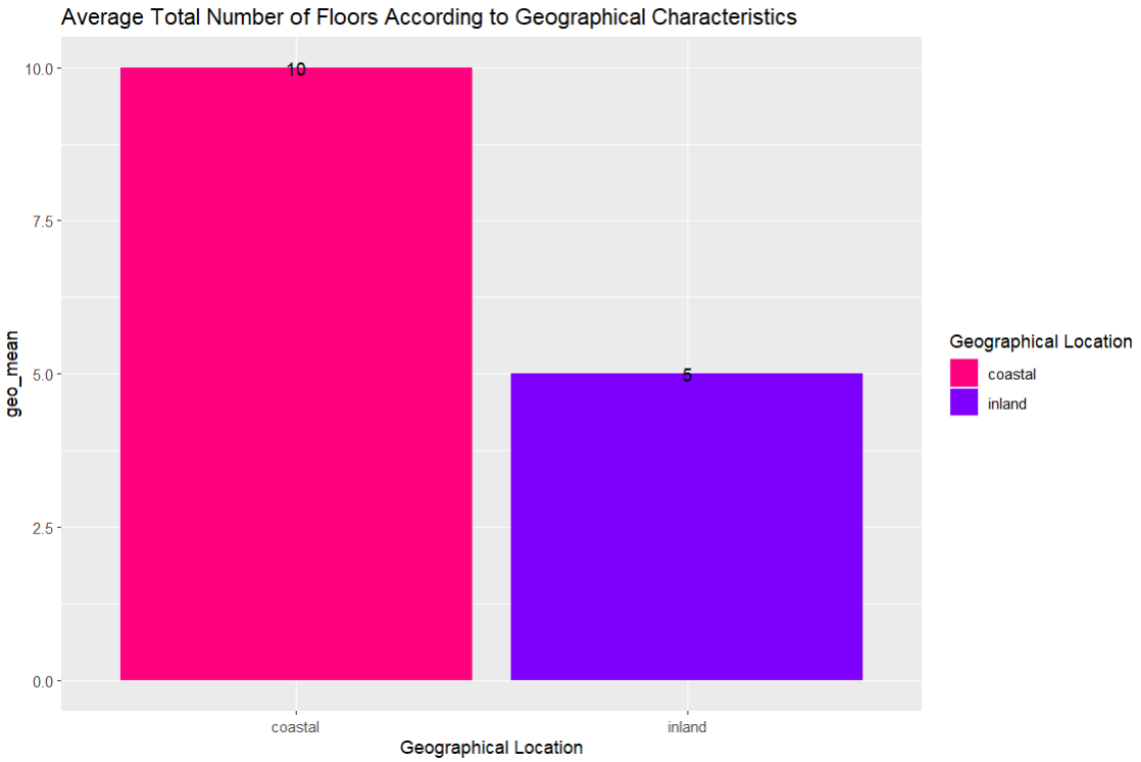

Figure 4-102: Average Total Number of Floors According to Geographical Characteristics

The bar plot shows that the housing buildings located at the coastal area is taller than the ones located within the inland by half. This could be due to the booming number of residents and the travel industry present at the coastal areas. Other than that, Mumbai as a coastal area has contributed as the population is deemed to be one of the biggest in India. Therefore, the demand for high-rise buildings is also necessary to fit in more people with safety precautions implemented while building.

4.5.9 Conclusion

The total number of floors for each building is affected by several factors present in the city that affects the architecture of the building. According to results, Mumbai tends to have a lot of high-rise buildings compared to other cities and this can be due to the localities present within the city. Mumbai consists of Bandra West, a locality that has significantly tall buildings as it is one of the most popular localities in the city for luxurious facilities and consists of a huge number of retail shops that attracts the crowd to reside there. Other than that, the strategic geographical location of coastal areas has increased the number of residents and tourists in visiting the areas. it is more convenient and space conserving to build the building upwards rather than sidewards as the spaces in coastal areas are rather limited due to the proximity kept from the shore while constructing the building.

On the other hand, Chennai who has the greatest number of short buildings among the cities present but has buildings overall taller than the ones in Kolkata, consists of the Velachery locality, which buildings are significantly short compared with others as it focuses more on providing commercial and residential needs rather than tourism. Its population is considered as moderate. Hence, the city can be more generous in terms of providing space to construct their properties sideways. Although the city is located in the coastal area as well, it might not affect the properties that greatly due to the main industry of the city.

4.6 Question 6: When and where do people usually start looking for houses?

4.6.1 Parse dates to day, month and year

Refer to:

Extra Feature 10: str_split_fixed()

Source Code:

```
house_data[c('Year', 'Month', 'Day')] <- str_split_fixed(house_data$`Posting Date`,'-', 3)
unique(house_data$Month)
unique(house_data$Year)
```

Figure 4-103: Parse dates to columns

The values of the Posting Date column are separated to 3 columns – Year, Month and Day with str_split_fixed() function.

Outcome:

Year	Month	Day
2022	05	18
2022	05	13
2022	05	16
2022	07	04
2022	05	09
2022	04	29
2022	06	21
2022	06	21
2022	06	07
2022	06	20
2022	05	23
2022	06	07
2022	05	14
2022	05	09
2022	05	05
2022	06	01
2022	05	17
2022	06	20
2022	06	09
2022	06	09
2022	07	02
2022	06	14
2022	06	15
2022	06	15

Figure 4-104: Year, Month, Day Columns

The columns consist of their respective values in the form of strings.

4.6.2 Analysis 6-1: Demand of houses over time

Analysis techniques used:

Analysis Technique	Reason
Data Manipulation	The number of properties per day is counted according to the posting date and stored in a data frame
Data Visualization	A line plot is made to illustrate the trend of the demands of houses

Source Code:

```
demand_houses <- data.frame(house_data %>% count(house_data$`Posting Date`))
demand_houses

ggplot(demand_houses, aes(x=house_data..Posting.Date., y=n, color=n)) +
  geom_line() +
  geom_smooth(method=lm) +
  scale_y_continuous(
    labels = function(x)
      format(x, big.mark = " ", scientific = FALSE)
  ) +
  theme_light() +
  theme(panel.grid = element_blank()) +
  labs(title="Demand of Houses Over Time") +
  xlab("Posting Date") +
  ylab("Number of Houses")
```

Figure 4-105: Manipulate and Visualize the Demand of House Over Time

The count of each posting dates from the house_data data frame are calculated with the count() function, which will be stored in the data frame named demand_houses. A line plot is plotted with ggplot() by having the posting date as the x-axis and n as the y-axis and color, geom_line() is implemented to construct the line according to the values inserted. The geom_smooth() function is used to illustrate the relationship between the two variables within the graphs using the linear method. The function scale_y_continuous() is used to specify the label names on the x-axis by passing a function that specifies the format within the argument. Other elements such as theme(), labs(), xlab() and ylab() are used to customize the design and of the plot so that it adheres to the purpose of plot of visualizing the demand of houses over time.

Plot:

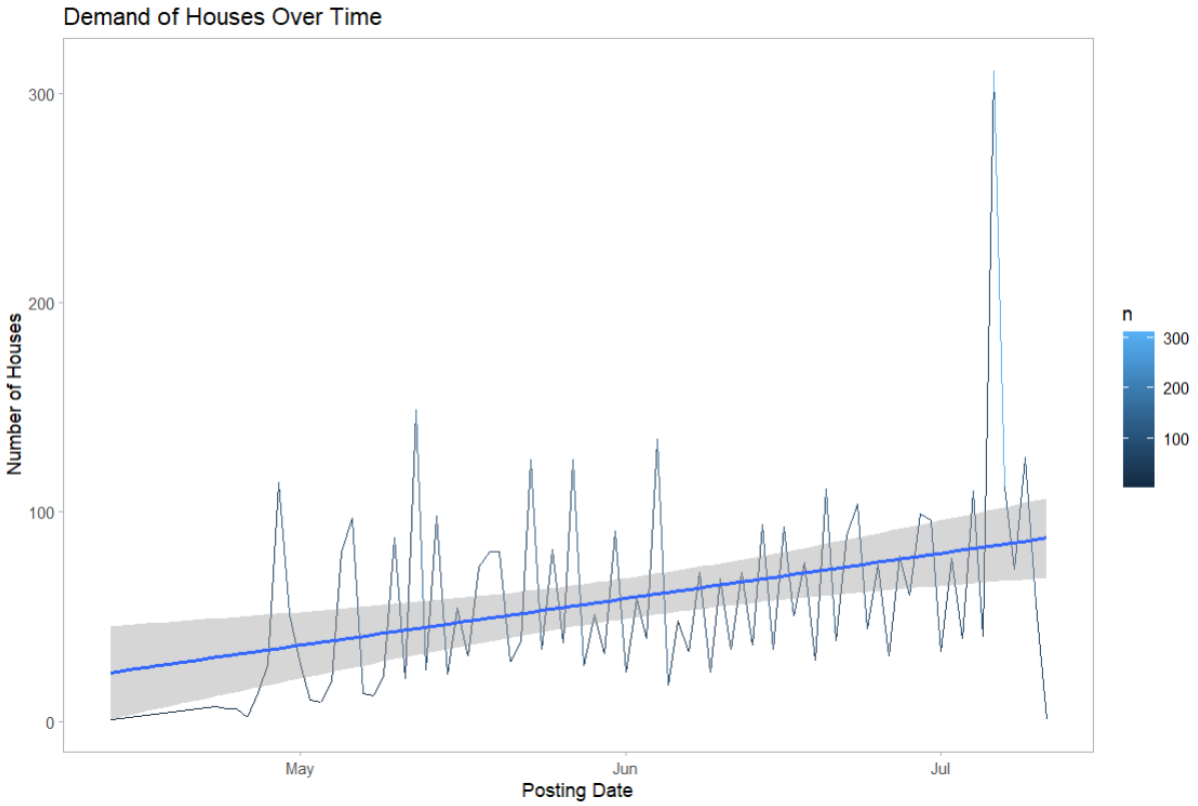

Figure 4-106: Demand of Houses Over Time

The line graph has illustrated that the demand of houses increases as the time goes by. The demand has reached its peak on the middle of July where the value has shot up to more than 300 houses per day. This can conclude the everlasting needs of humans in finding a suitable property for them to stay as there are many factors behind that might cause some action. Other than that, people might utilize the number of holidays given to them to move into the house while nearing the end of the year (Johnson M. , 2017). Hence, providing them with a good opportunity for house hunting.

4.6.3 Analysis 6-2: Rental price of houses over time

Analysis techniques used:

Analysis Technique	Reason
Data Manipulation	The mean rental price per day is counted according to the posting date and stored in a data frame
Data Visualization	A line plot is made to illustrate the trend of the rental prices of houses

Source Code:

```
time_rent_mean <- house_data %>%
  filter(`Posting Date` > '2022-04-13') %>%
  group_by(`Posting Date`) %>%
  summarise(rent_mean=round(mean(`Rent Price`), 2))
time_rent_mean

ggplot(time_rent_mean, aes(x=`Posting Date`, y=rent_mean, color=rent_mean)) +
  geom_line() +
  geom_smooth(method = lm, color = 'orange') +
  scale_y_continuous(
    labels = function(x)
      format(x, big.mark = " ", scientific = FALSE)
  ) +
  scale_colour_gradient(low = "blue", high = "orange") +
  theme_light() +
  theme(panel.grid = element_blank()) +
  labs(title="Mean Rental Prices of Houses Over Time")
```

Figure 4-107: Manipulate and Visualize the Mean Rental Prices of Houses Over Time

To calculate the mean of the rent price for each posting date, the posting date after 2022-04-03 is selected due to the outlier produced by the date 2022-04-03 itself. The mean() function is used to calculate the mean of the rent price while rounding it up to two decimal by the round() function within the summarise() function so that the operation can be transformed into a new column, it is grouped by the posting dates using the group_by() function. A line plot is plotted with ggplot() by having the posting date as the x-axis and rent_mean as the y-axis and color, geom_line() is implemented to construct the line according to the values inserted. The geom_smooth() function is used to determine the relationship between the two variables within the graphs using the linear method. The function scale_y_continuous() is used to specify the label names on the x-axis by passing a function that specifies the format within the argument. Other elements such as theme(), labs(), xlab() and ylab() are used to customize the

design and of the plot so that it adheres to the purpose of plot of visualizing the demand of houses over time.Lastly, the scale_color_gradient () is used by determining the color shown when its value is low, which is blue and when its value is high, which is orange in a gradient form.

Plot:

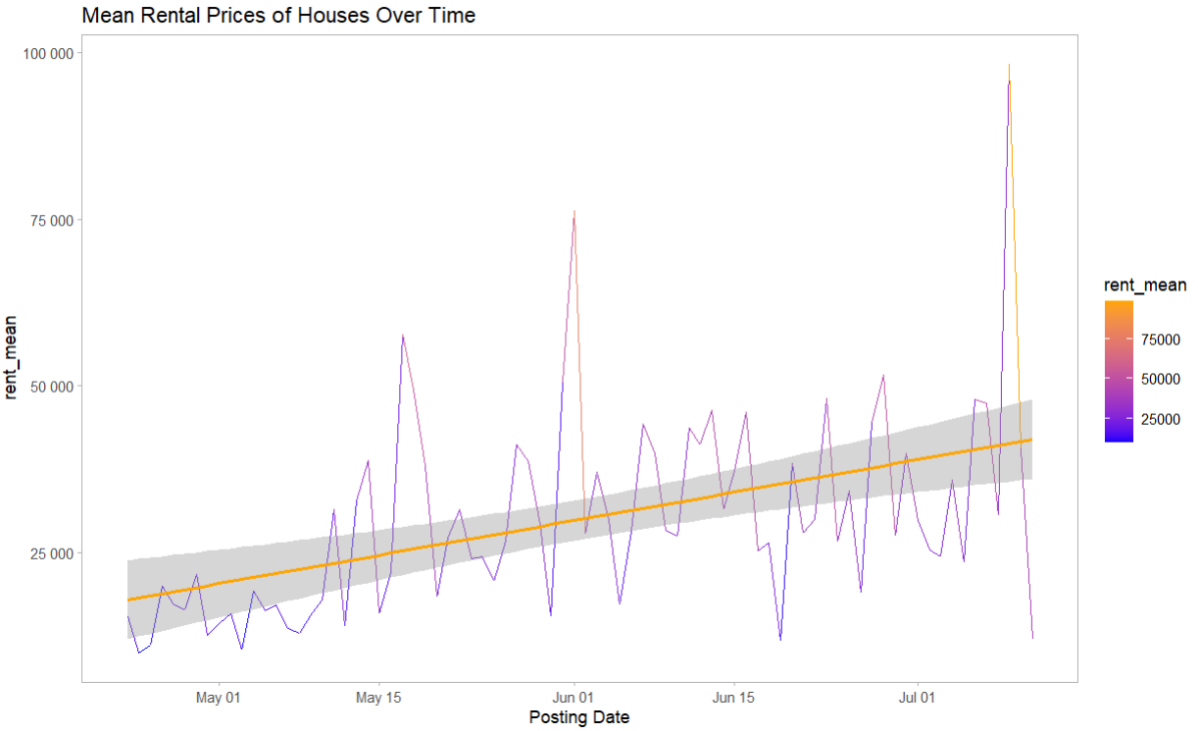

Figure 4-108: Mean Rental Prices of Houses Over Time

According to the time series line plot, the mean of the rental prices increases as it approaches towards the end of the year. The price has reached its peak at the middle of July, and it was at its lowest at the middle of June. This could be related to the results obtained at Analysis 6-1, where the demand of houses rises at the same period of time in a similar trend. House owners are most likely to raise the rent of the house due to the high demand present so that it can attract better tenants to reside within their property.

4.6.4 Analysis 6-3: Determine the month where bachelors usually start finding houses

Analysis techniques used:

Analysis Technique	Reason
Data Exploration	The number of rows of bachelor demand is explored according to month
Data Manipulation	The month labels and the total number of bachelors for each month are placed into a data frame
Data Visualization	A pie chart is made to visualize the bachelors demand according to month

Source Code:

```
bachelor_time <- function(month)
{
  nrow(house_data[(house_data$`Preferred Tenants` == "bachelors" |
                   house_data$`Preferred Tenants`== "bachelors/family")
                & house_data$Month == month,])
}

bachelor_april <- bachelor_time('04')
bachelor_may <- bachelor_time('05')
bachelor_june <- bachelor_time('06')
bachelor_july <- bachelor_time('07')

bachelor_time_demand <- c(bachelor_april, bachelor_may, bachelor_june, bachelor_july)
month <- c('April','May', 'June', 'July')

bachelor_month <- data.frame(month, bachelor_time_demand)

ggplot(bachelor_month, aes(x="", y=bachelor_time_demand, fill=month)) +
  geom_col() +
  geom_text(aes(label=bachelor_time_demand),
            position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  labs(title="Distribution of Bachelors Demand According to Month")
```

Figure 4-109: Explore, Manipulate and Visualize the Bachelors' Demand According to Month

The number of rows is calculated based on the bachelors and the month entered, which is defined as the bachelor_time() function. The function is called to four variables to calculate the number of bachelors renting houses for that particular month. The values and labels are stored into two separate vectors – bachelor_time_demand and month. Both vectors are combined to form a data frame named bachelor_month. Lastly, a pie chart is plotted with ggplot() with bachelor_month as the data, bachelor_time_demand as the y axis and month as

fill with using geom_col so that the heights of the bars represent the value. The labels are added by implementing geom_text() with bachelor_time_demand as the label stacked on the pie chart. The initial state of the chart is a bar chart. Hence, the use of coord_polar() helps to convert the stacked bar chart using polar coordinates to form a pie chart. A title – Distribution of Bachelors Demand According to Month is added with labs().

Plot:

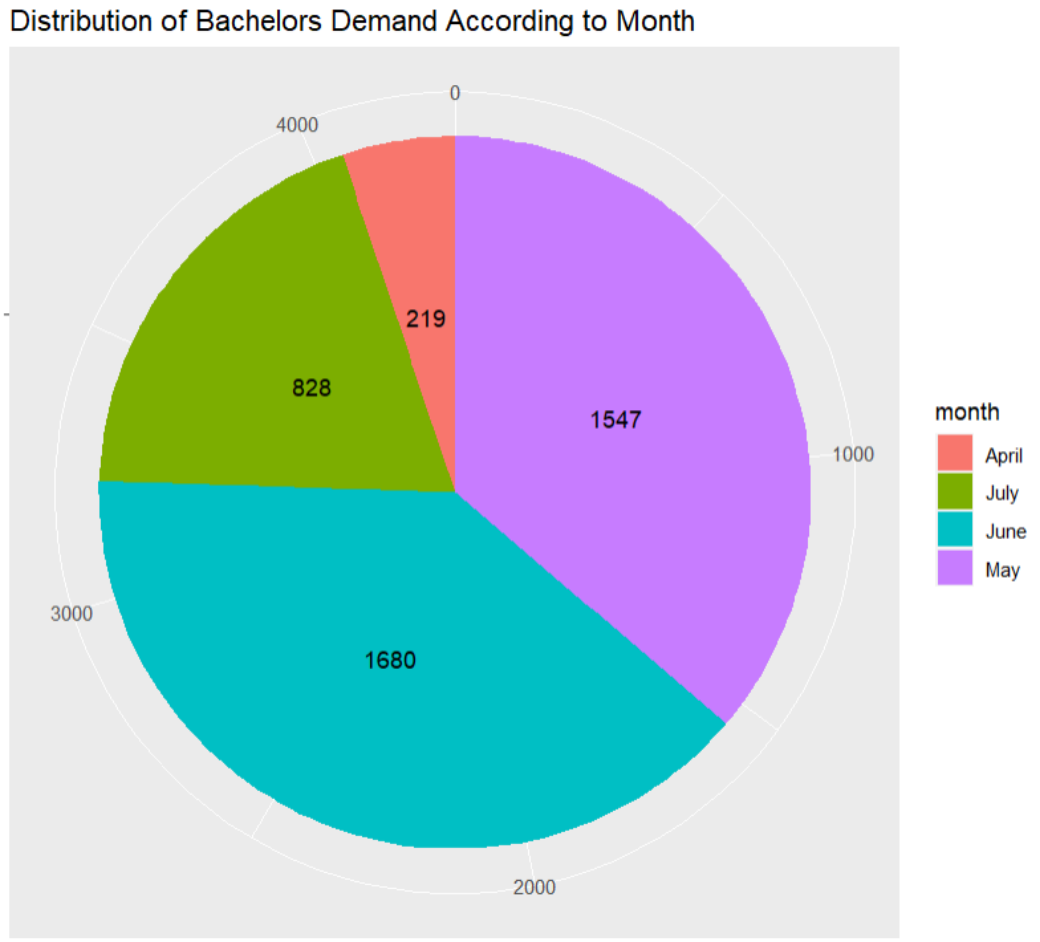

Figure 4-110: Distribution of Bachelors' Demand According to Month

The pie chart has illustrated that the month of June has the highest demand for Bachelors, followed by May with the difference of only 133 bachelors. April tends to be the least in terms of demand. Bachelors might grasp the opportunity in renting a house as they are having the upper hand compared to families. They might assume families are less likely to move out during the middle of the year as they do not wish to disturb the studies their schooling children. Plus, June is one of the busiest times of the year for work progression, which adds up to more burden compared to bachelors.

4.6.5 Analysis 6-4: Determine the month where families usually start finding houses

Analysis techniques used:

Analysis Technique	Reason
Data Exploration	The number of rows of family demand is explored according to month
Data Manipulation	The month labels and the total number of families for each month are placed into a data frame
Data Visualization	A radial chart is made to visualize the families demand according to month

Source Code:

```
family_time <- function(month)
{
  nrow(house_data[(house_data$`Preferred Tenants` == "family" |
                     house_data$`Preferred Tenants`== "bachelors/family")
                  & house_data$Month == month,])
}

family_april <- family_time('04')
family_may <- family_time('05')
family_june <- family_time('06')
family_july <- family_time('07')

family_time_demand <- c(family_april, family_may, family_june, family_july)
month <- c('April','May', 'June', 'July')

family_month <- data.frame(month, family_time_demand)

ggplot(family_month) +
  geom_hline(
    aes(yintercept = y),
    data.frame(y = c(0:1) * 1000),
    color= "lightgrey") +
  geom_col(aes(
    x = reorder(str_wrap(month, 4), month),
    y = family_time_demand,
    fill = month),
    position = "dodge2",
    show.legend = TRUE,
    alpha= .9) +
  geom_segment(aes(
    x = reorder(str_wrap(month, 4), month),
    y = 0,
    xend = reorder(str_wrap(month, 4), month),
    yend = 2000),
    linetype = "dashed",
    color = "gray12") +
  coord_polar() +
  labs(title = "Distribution of Families Demand According to Month") +
  ylab("Number of Families") +
  xlab("Month")
```

Figure 4-111: Explore, Manipulate and Visualize the Families Demand

The number of rows is calculated based on the bachelors and the month entered, which is defined as the family_time() function. The function is called to four variables to calculate the number of families renting houses for that particular month. The values and labels are stored into two separate vectors – family_time_demand and month. Both vectors are combined to form a data frame named family_month. A radial plot is then plotted with ggplot() function, which will be elaborated in the extra features section.

Radial Plot

Extra Feature 13: Radial Plot

Plot:

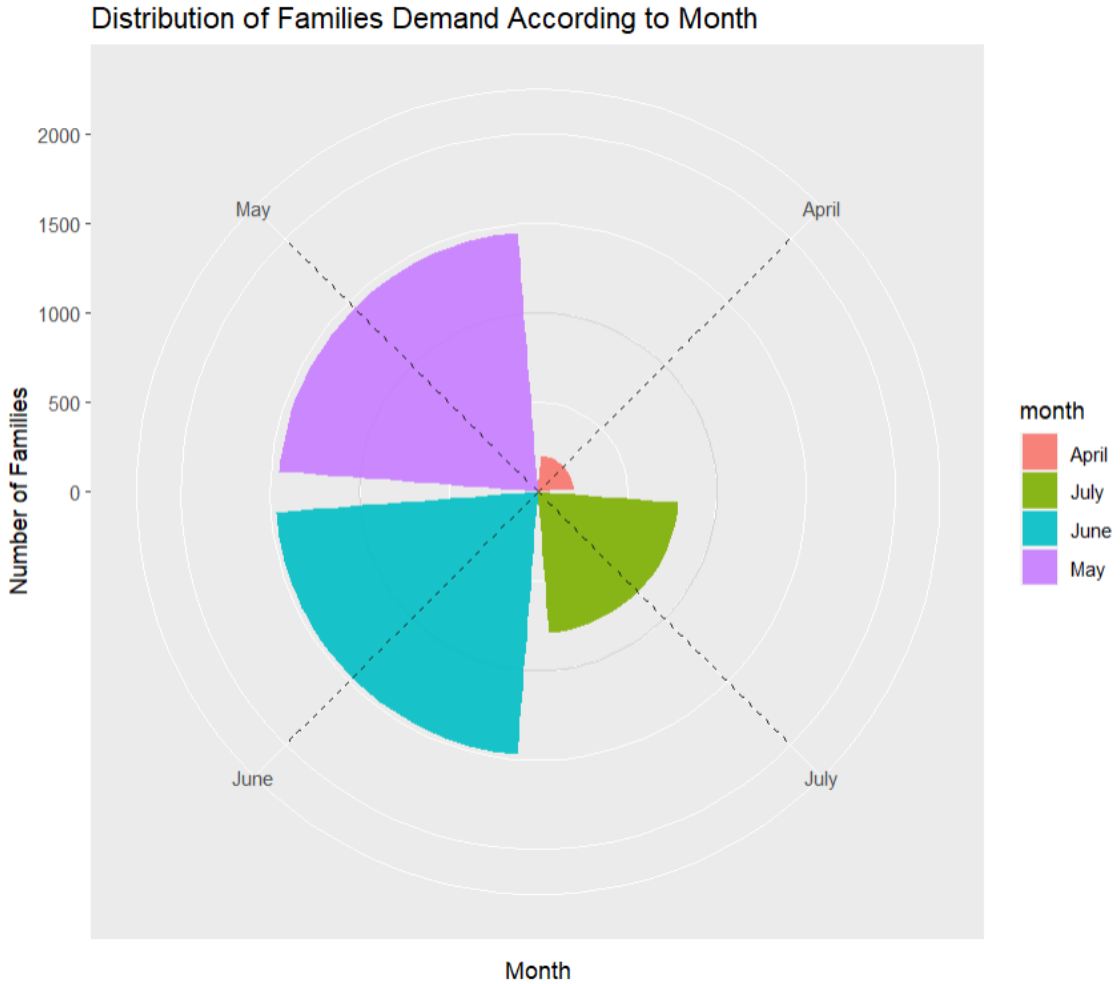

Figure 4-112: Distribution of Families Demand According to Month

Based on the radial plot, the demand of houses in June seems to be higher than most of the other months and has only a slight difference with the demand during May. From the family's perspective, June is a more productive timing for them to earn extra or to increase their income due to the heavier workload experienced before entering autumn. Hence, more funds can be generated for the family to search for a new unit to rent. However, the scheduled time to move out can be customized and planned by the family members themselves.

4.6.6 Analysis 6-5: Determine the city that has the most demand during April

Analysis techniques used:

Analysis Technique	Reason
Data Manipulation	The data is filtered, grouped by, counted, summed, arranged and transformed to percentage
Data Visualization	A pie chart is illustrated to view the demand of properties for each city in April 2022

Source Code:

```
#Transform to percentage
city_apr_dist <- house_data %>%
  filter(Month == "04") %>%
  group_by(`City Located`) %>%
  count() %>%
  ungroup() %>%
  mutate(city_apr = `n` / sum(`n`)) %>%
  arrange(city_apr) %>%
  mutate(labels = scales::percent(city_apr))

city_apr_dist

ggplot(city_apr_dist, aes(x="", y = city_apr, fill = `City Located`)) +
  geom_col() +
  geom_text(aes(label = labels),
            position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  labs(title="Demand of Properties for Each City In April 2022")
```

Figure 4-113: Demand of Properties for Each City in April 2022

The city_apr_dist data frame is created according to the data present in the house_data data frame, where the rows that represents houses at the month of April is filtered with filter(). Then the count of the data is grouped by according to the City Located column with group_by(). After calculating the count, the data is ungrouped to form independent categories. The values are calculated as percentage values by dividing n with the sum of n and it is stored in the city_apr column with the values arranged in ascending order. Another column is formed using the mutate() function to transform the decimals obtained in the city_apr column into the percentage format. A pie chart is formed with ggplot() to properly view the distribution of the demand of houses present in each city for the month of April 2022., with city_apr_dist as the

data, city_apr as the y axis and City Located as fill with using geom_col so that the heights of the bars represent the value. The labels are added by implementing geom_text() with labels as the label stacked on the pie chart. The initial state of the chart is a bar chart. Hence, the use of coord_polar() helps to convert the stacked bar chart using polar coordinates to form a pie chart. A title – Demand of Properties for Each City in April 2022 is added with labs().

Plot:

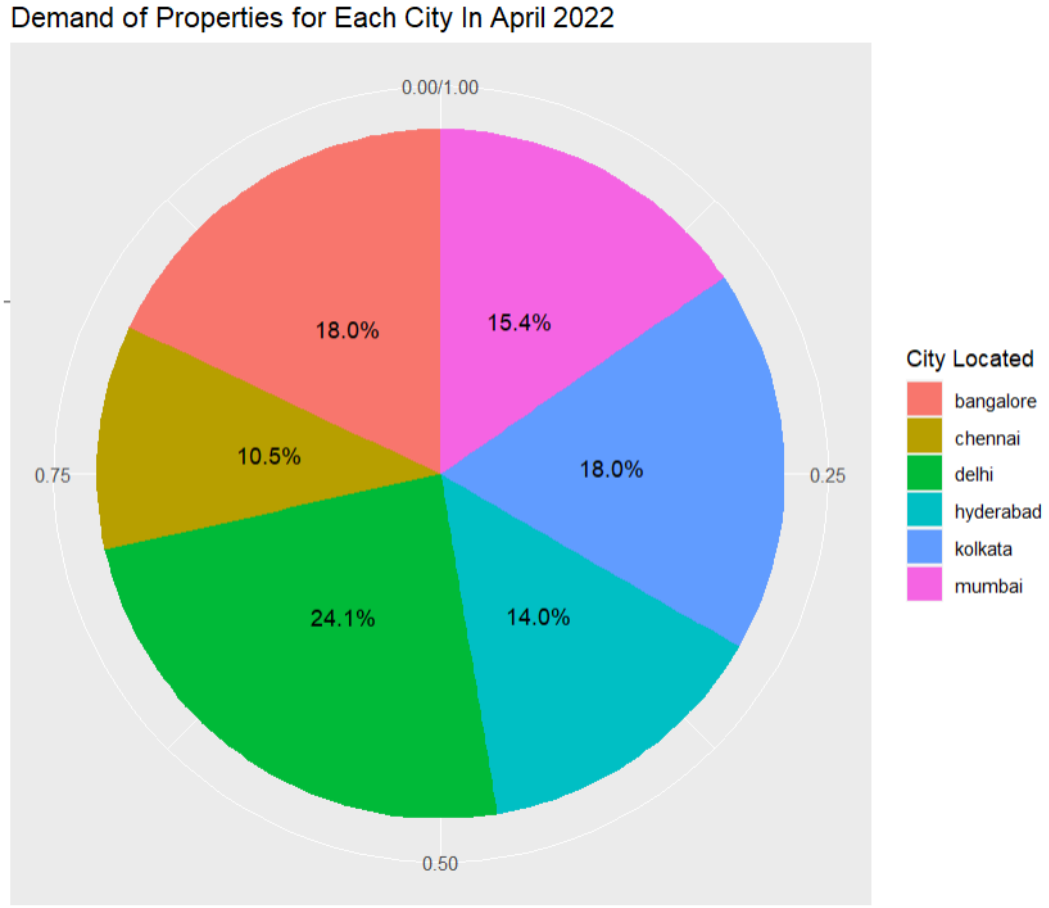

Figure 4-114: Demand of Properties for Each City in April 2022

The pie chart states that Delhi has the highest demand of properties during April 2022 with the amount of 24.1% whereas Chennai has the lowest demand of properties during the month. The demand of properties is analyzed for house owners to understand the demands of people so that the house can be posted out at a good timing to attract more tenants and increase the likelihood to rent the unit.

4.6.7 Analysis 6-6: Determine the city that has the most demand during May

Analysis techniques used:

Analysis Technique	Reason
Data Manipulation	The data is filtered, grouped by, counted, summed, arranged and transformed to percentage
Data Visualization	A pie chart is illustrated to view the demand of properties for each city in May 2022

Source Code:

```
#Transform to percentage
city_may_dist <- house_data %>%
  filter(Month == "05") %>%
  group_by(`City Located`) %>%
  count() %>%
  ungroup() %>%
  mutate(city_may = `n` / sum(`n`)) %>%
  arrange(city_may) %>%
  mutate(labels = scales::percent(city_may))

ggplot(city_may_dist, aes(x="", y = city_may, fill = `City Located`)) +
  geom_col() +
  geom_text(aes(label = labels),
            position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  labs(title="Demand of Properties for Each City In May 2022")
```

Figure 4-115: Demand of Properties for Each City in May 2022

The city_may_dist data frame is created according to the data present in the house_data data frame, where the rows that represents houses at the month of May is filtered with filter(). Then the count of the data is grouped by according to the City Located column with group_by(). After calculating the count, the data is ungrouped to form independent categories. The values are calculated as percentage values by dividing n with the sum of n and it is stored in the city_may column with the values arranged in ascending order. Another column is formed using the mutate() function to transform the decimals obtained in the city_may column into the percentage format. A pie chart is formed with ggplot() to properly view the distribution of the demand of houses present in each city for the month of May 2022., with city_may_dist as the data, city_may as the y axis and City Located as fill with using geom_col so that the heights of

the bars represent the value. A title – Demand of Properties for Each City in May 2022 is added with labs().

Plot:

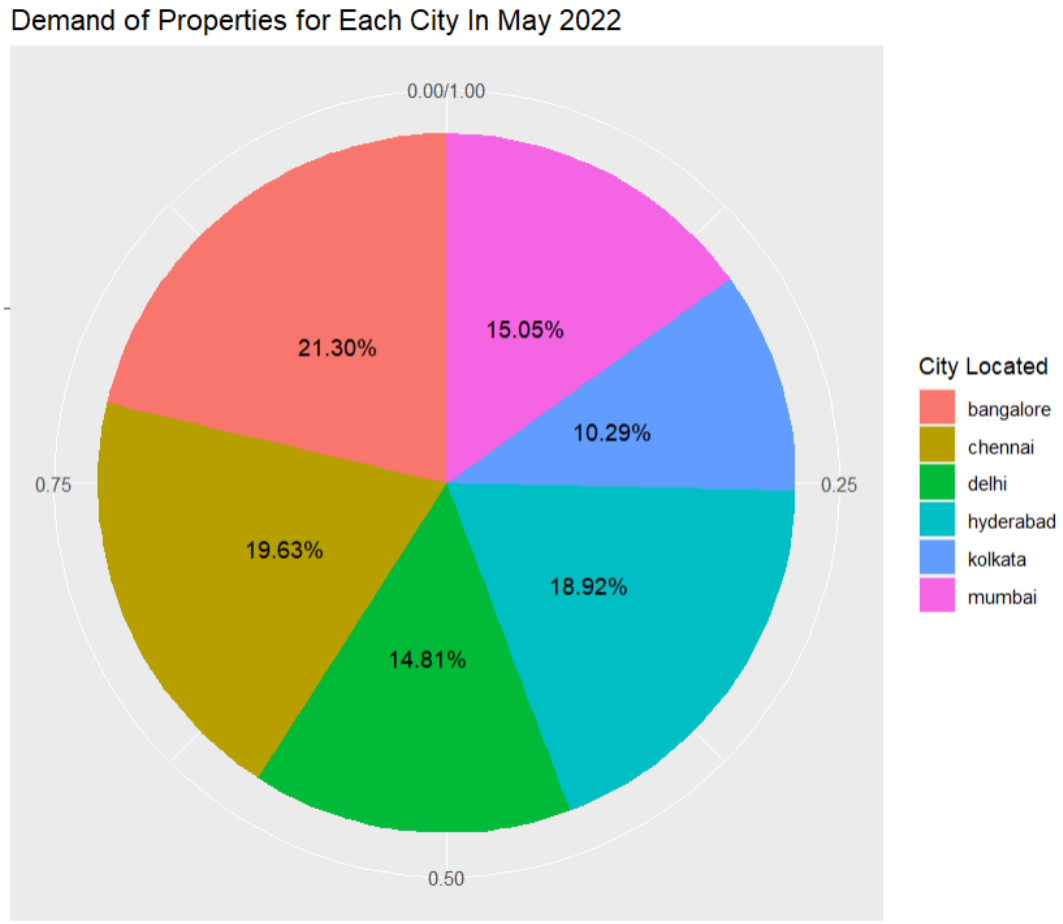

Figure 4-116: Demand of Properties for Each City in May 2022

According to the pie chart, Bangalore has the highest demand of properties during May 2022. On the other hand, houses in Kolkata do not seem to be popular within that month.

4.6.8 Analysis 6-7: Determine the city that has the most demand during June

Analysis techniques used:

Analysis Technique	Reason
Data Manipulation	The data is filtered, grouped by, counted, summed, arranged and transformed to percentage
Data Visualization	A tree map is illustrated to view the demand of properties for each city in June 2022

Source Code:

```
#Transform to percentage
city_june_dist <- house_data %>%
  filter(Month == "06") %>%
  group_by(`City Located`) %>%
  count() %>%
  ungroup() %>%
  mutate(city_june = `n` / sum(`n`)) %>%
  arrange(city_june) %>%
  mutate(labels = scales::percent(city_june))

#Plot tree map
ggplot(city_june_dist, aes(area = n, fill = n, label = paste(`City Located`,
                                                             labels, sep = "\n"))) +
  geom_treemap() +
  geom_treemap_text(colour = "white",
                    place = "centre",
                    size = 15) +
  scale_fill_gradient(low = "orange", high = "purple") +
  ggtitle("Demand of Properties for Each City In June 2022") +
  labs(fill = "Total Properties")
```

Figure 4-117: Demand of Properties for Each City in May 2022

The transformation of data is similar to the method present in Analysis 6-5 and Analysis 6-6, with the difference of variable names and selected filters. The data frame variable is altered to city_june_dist and the column to store the decimal values are assigned as city_june. The rows with June as the month will be filtered to visualize the pie chart. A tree map is plotted with ggplot() and geom_treemap().

Plot:

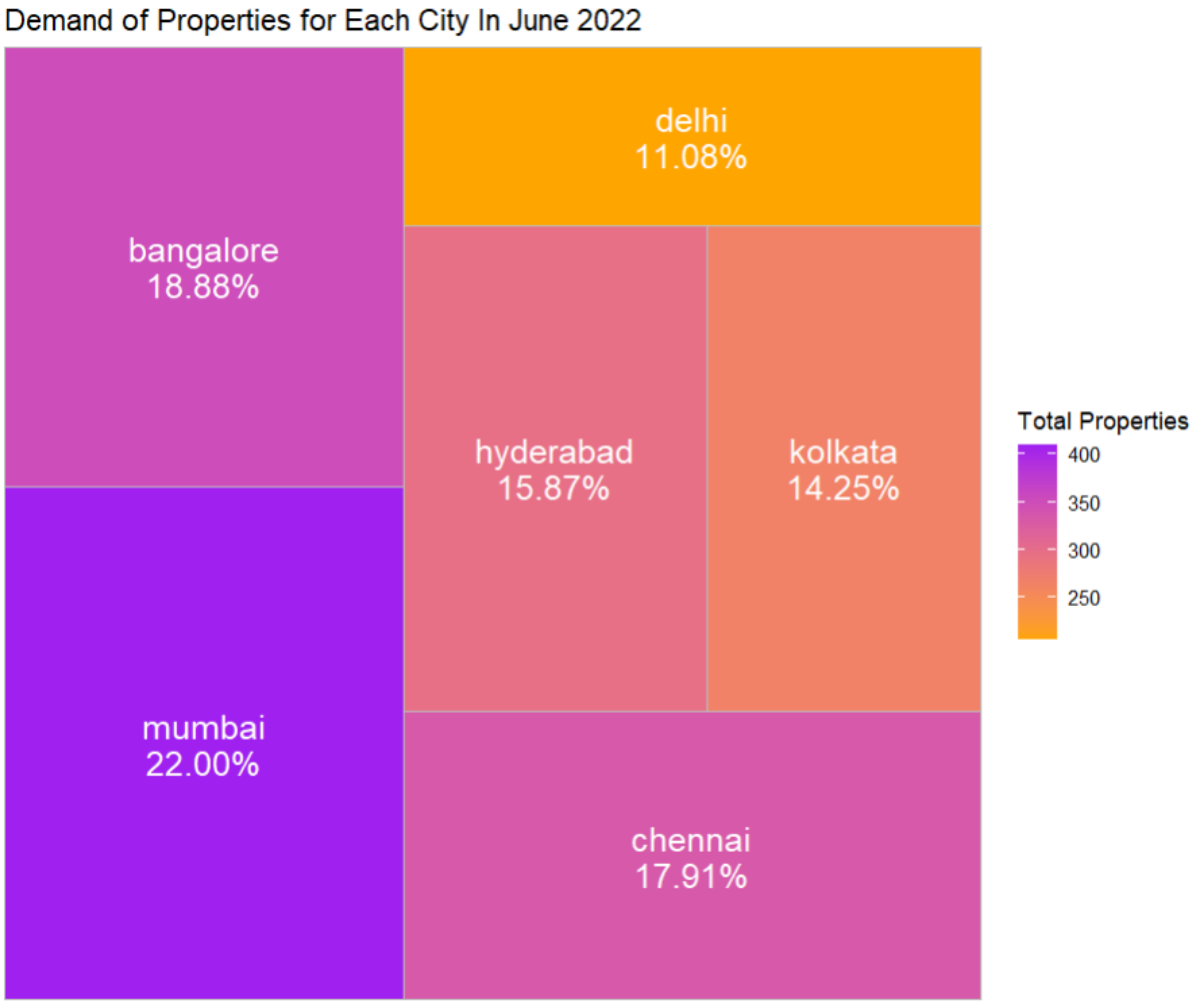

Figure 4-118: Demand of Properties for Each City in June 2022

The tree map illustrates that Mumbai has the highest demand of houses during the month of June 2022 whereas Delhi has the lowest demand of houses at the period of time.

4.6.9 Analysis 6-8: Determine the city that has the most demand during July

Analysis techniques used:

Analysis Technique	Reason
Data Manipulation	The data is filtered, grouped by, counted, summed, arranged and transformed to percentage
Data Visualization	A tree map is illustrated to view the demand of properties for each city in July 2022

Source Code:

```
#Transform to percentage
city_july_dist <- house_data %>%
  filter(Month == "07") %>%
  group_by(`City Located`) %>%
  count() %>%
  ungroup() %>%
  mutate(city_jul = `n` / sum(`n`)) %>%
  arrange(city_jul) %>%
  mutate(labels = scales::percent(city_jul))

#Plot tree map
ggplot(city_july_dist, aes(area = n, fill = n, label = paste(`City Located`,
                                                             labels, sep = "\n"))) +
  geom_treemap() +
  geom_treemap_text(colour = "white",
                    place = "centre",
                    size = 15) +
  scale_fill_gradient(low = "turquoise", high = "gold") +
  ggtitle("Demand of Properties for Each City In July 2022") +
  labs(fill = "Total Properties")
```

Figure 4-119: Demand of Properties for Each City in July 2022

The transformation of data is similar to the method present in Analysis 6-5, Analysis 6-6 and Analysis 6-7, with the difference of variable names and selected filters. The data frame variable is altered to city_july_dist and the column to store the decimal values are assigned as city_july. The rows with July as the month will be filtered to visualize the pie chart. A tree map is plotted with ggplot() and geom_treemap().

Plot:

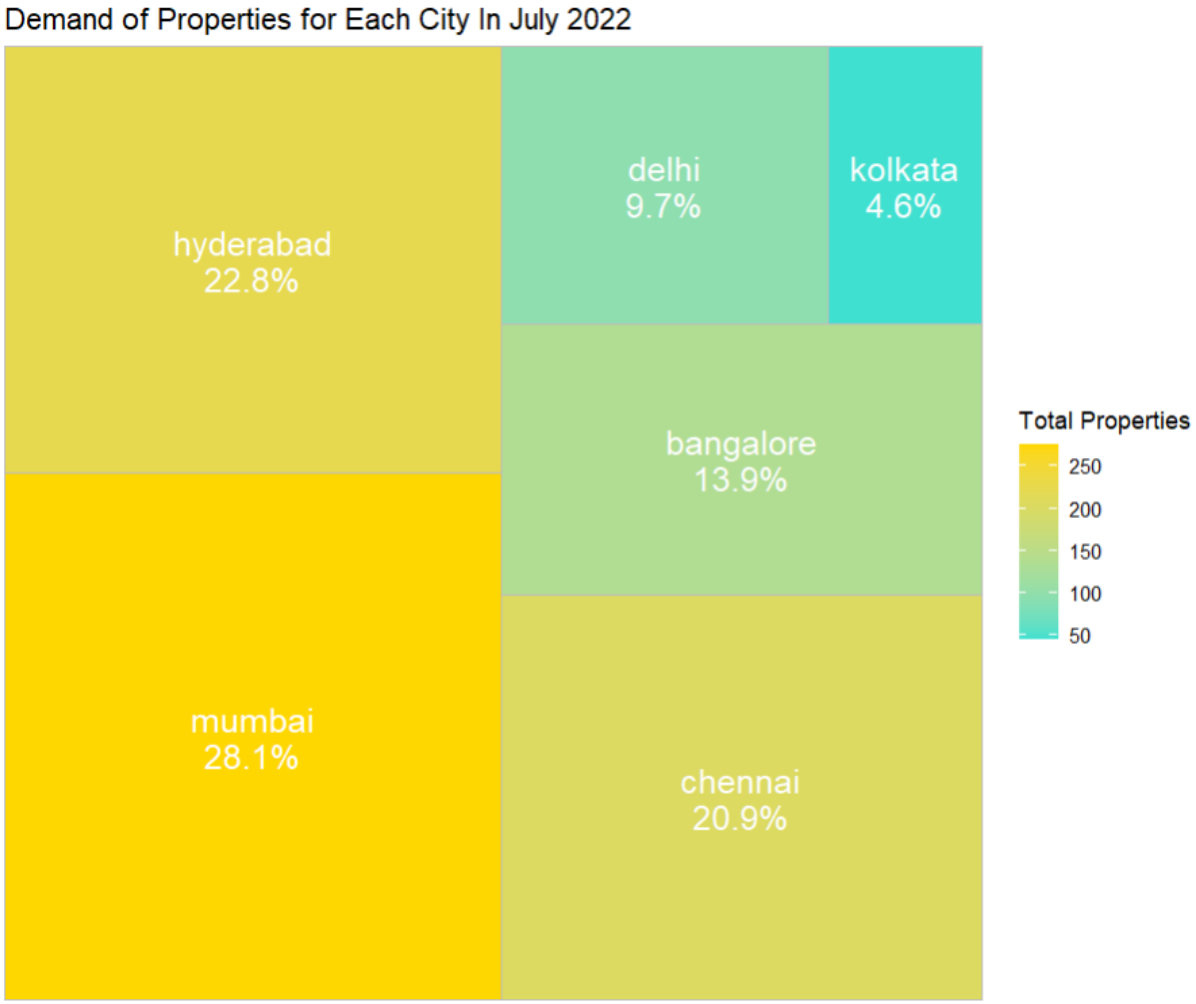

Figure 4-120: Demand of Properties for Each City in July 2022

Mumbai has the biggest piece within the tree map, with the percentage of 28.1% which makes the city with the highest demand of properties in July 2022. On the other hand, Kolkata has the smallest piece of the tree map, which is the city with the least property demand within the month.

4.6.10 Analysis 6-9: Determine the price category of houses according to month

Analysis techniques used:

Analysis Techniques	Reason
Data Manipulation	The rounded mean rent price is calculated and is transformed into a new column while being grouped by the city located
Data Visualization	A stacked bar chart is visualized to illustrate the distribution of the mean of houses for each city within four months

Source Code:

```
month_houses <- house_data %>%
  group_by(`City Located`, Month) %>%
  dplyr::summarise(city_month_mean=round(mean(`Rent Price`)), .groups = 'drop')
month_houses

ggplot(month_houses, aes(x=Month, fill=`City Located`)) +
  geom_bar(aes(weight=city_month_mean), position="stack") +
  geom_text(position = "stack", aes(Month,city_month_mean,label=city_month_mean), size = 5) +
  labs(title="Distribution of The Mean of Houses for Each City Within Four Months", y="Mean Rent")
```

Figure 4-121: Manipulate and Visualize the Distribution of the Mean of Houses for Each City Within Four Months

The columns from the house_data data frame is initiated to extract the columns present within the data frame to create a new data frame named month_houses. The mean of the rent price is rounded up to 2 decimal point and it is transformed into the column named city_month_mean with the summarise() function, the .groups argument is placed as drop to prevent errors while visualizing the chart. A bar plot is made through ggplot() with the use of month_houses as data, Month as x-axis and City Located as the fill. To make the bars and labels stack accordingly, the position argument has to be passed with the value “stack” in the geom_bar() and geom_text() function. The labels are taken from the city_month_mean column and the title is set as Distribution of The Mean of Houses for Each City Within Four Months with the label of the y-axis named as Mean Rent.

Plot:

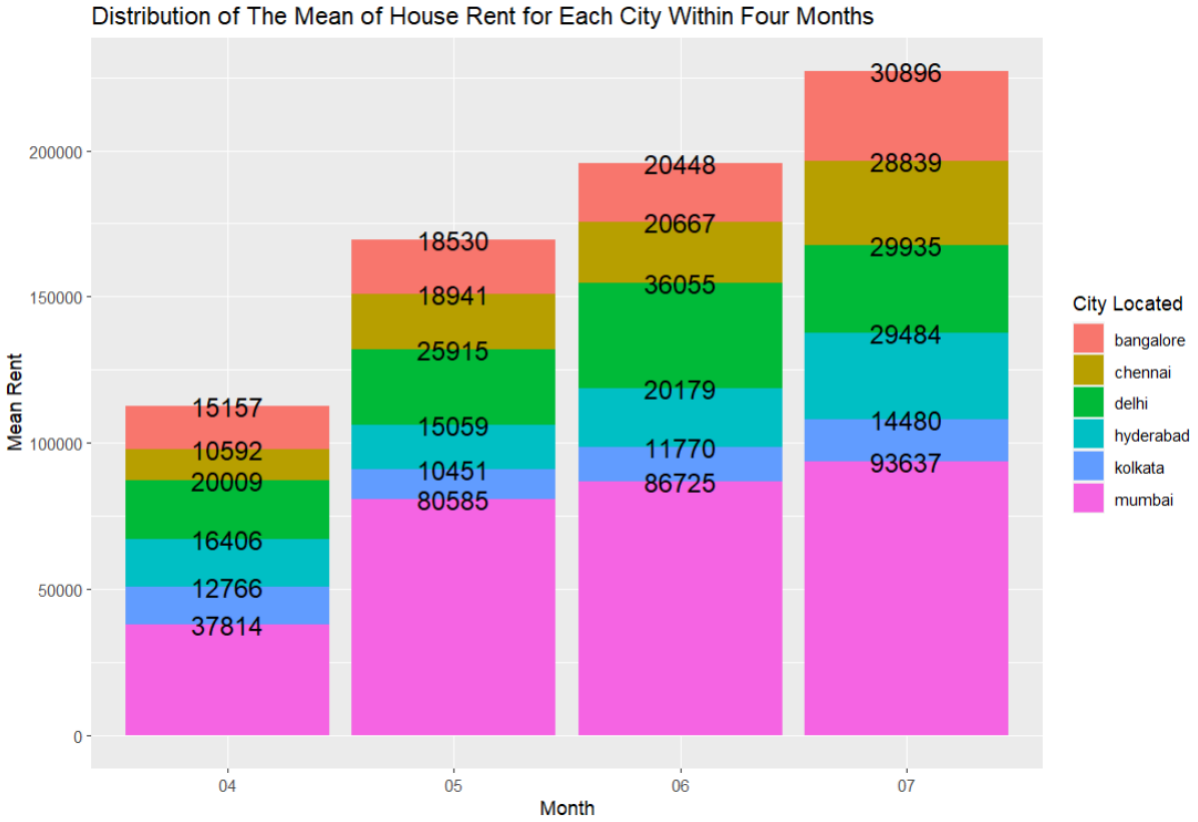

Figure 4-122: Distribution of the Mean of House Rent for Each City Within Four Months

The house rent gradually increases as the months go by and the pattern is followed by most of the cities except for Delhi, where a slight decline in the mean house rent is spotted during July 2022. Mumbai has the highest value of rent among all of the cities within the four months whereas Kolkata scores the lowest among all. This also shows that most of the potential tenants' behaviour are similar in terms of looking for houses as the demand of the houses goes up when it approaches to the end of the year within most cities and that contributes to the increase of house rent.

4.6.11 Conclusion

To conclude, the demand of houses varies upon the timing of the year as the data proves that the demand of houses increases when the month increases, and it will most likely carry the same trend during the next year. The demand of houses reaches its peak in July 2022, whereas it is at its lowest in April 2022, which enables the rent price to inherit the same trend as the demand affects the rent price set by house owners in order to grab the attention of potential tenants. Through the analysis done, it can also be concluded that marital status does not bring a big impact in selecting a timing to find houses as both sides tend to have different perspectives and ways to tackle the challenges they face while hunting for houses to rent. This can be proven by looking into the high demands for properties in both sides during July and its lowest in May. The condition and culture within each city are related to the demand of houses from time to time. The analysis results shows that the city with the highest demand of houses changes each month, it can be presented as following:

Month	**Highest Demand City**	**Lowest Demand City**
April	Delhi	Chennai
May	Bangalore	Kolkata
June	Mumbai	Delhi
July	Mumbai	Kolkata

Table 4-2: High and Low Demand Cities

The overall increase of rent price is contributed by most of the increase of the rent price that is happening within most of the cities. However, there are still some cities that face the decrease of rental price when approaches towards the end of the year, where it can be affected by factors such as monsoon and economic factors.

5.0 Extra Features

5.1 Extra Feature 1: par()

Source Code:

```
#View distribution of all numerical columns (Data Exploration)

temp_house_data <- data.frame(house_data$`BHK In Unit`,house_data$`Unit Size`,
                              house_data$`Rent Price`, house_data$`Number of Bathroom`)
head(temp_house_data)

par(mfrow = c(2,2))
loop_vector <- 1:4

for (i in loop_vector)
{
  x <- temp_house_data[,i]

  hist(x, main = paste("Distribution of column", i),
       xlab = names(temp_house_data[ ,i]))
}
```

Figure 5-1: par() Function

The par() function is used to determine or query parameters in graphics by splitting the frame into the desired grid, adding margins to the plot, perform changed to the color of the frame background or to construct multiple plots in one go (GeeksforGeeks, 2021). For the current scenario, the columns – BHK In Unit, Unit Size, Rent Price and Number of Bathroom are placed into one data frame. The par() function is used to divide the frame present into 2 x 2 grids through using the mfrow argument. A loop is then implemented to plot the histograms to view the distributions of each column.

Plot:

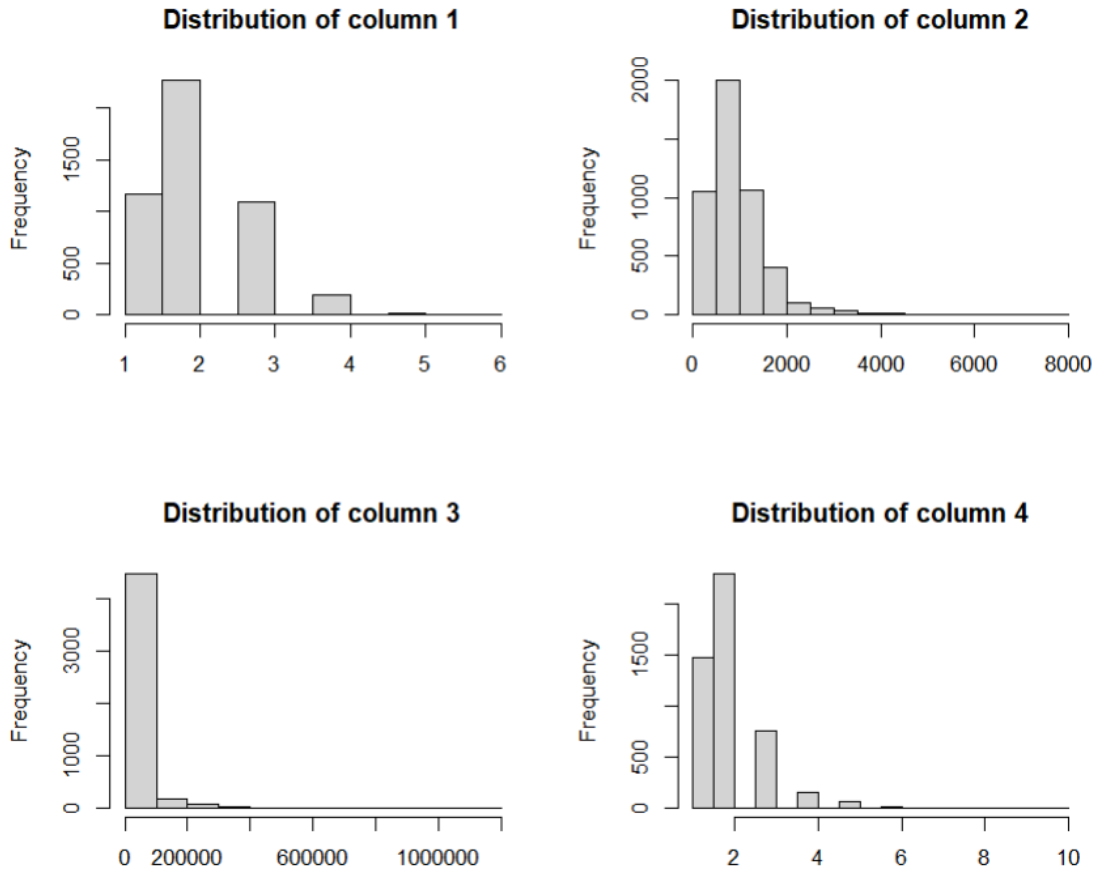

Figure 5-2: Distribution of the Columns

The plot is then visualized according to a 2 x 2 manner, the result shows that most of the columns after cleaning and manipulation are skewed to the left side.

Reason:

The function is used to ease the visualizations of similar plots as the same type of plots can be replicated if the function is utilized. Other than that, it eases decision makers to perform comparison between the distributions present for each column, which might potentially induce a relationship between the plots.

5.2 Extra Feature 2: duplicated()

Source Code:

```
#Finding duplicate values
duplicated(house_data)
```

Figure 5-3: duplicated() Function

The duplicated() function is used to determine duplicates of elements present within a vector or a data frame. The duplicates are scanned through by passing the house_data data frame into the duplicated() function and the system starts searching for duplicated inside the data frame.

Outcome:

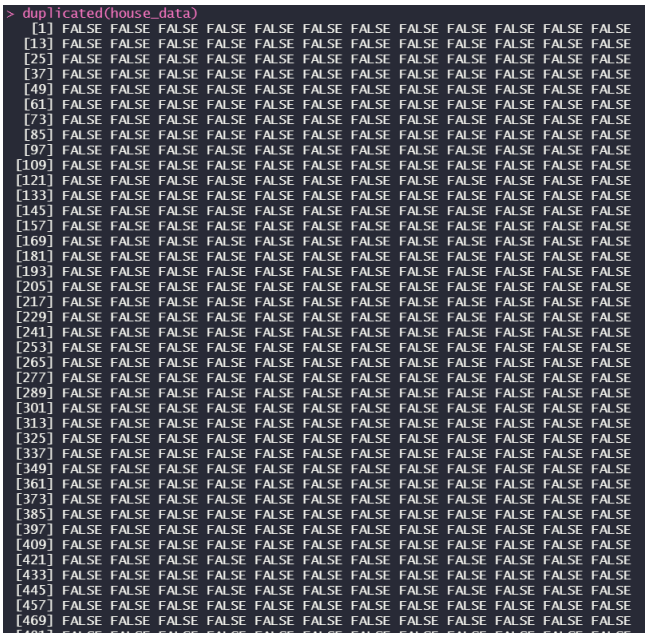

Figure 5-4: No Duplicates

Based on the outcome, no duplicates were found as it returns false for all rows. If a value is duplicated, it will return true.

Reason:

The duplicate() function can come in handy while data cleaning as it can determine duplicated values to be removed so that the outcome of the visualizations will not be affected.

5.3 Extra Feature 3: gsub()

Source Code:

```
#Remove unnecessary keywords such as "area" and "contact"
house_data$`Type of Area` <- gsub(' area','',as.character(house_data$`Type of Area`))
house_data$`Contact Person` <- gsub('contact ','',as.character(house_data$`Contact Person`))
```

Figure 5-5: gsub() function

The gsub() function is handy for string substitution as it replaces the string present in a vector or a data frame or to substitute the desired string. For this scenario, the patterns of their respective strings to be replaced such as ‘ area’ and ‘contact ’ are replaced with an empty string so that the specific part of the string can be removed from the column specified.

Outcome:

Type of Area	Contact Person
super	owner
super	owner
super	owner
super	owner
carpet	owner
super	owner
super	agent

Figure 5-6: Replaced Columns

The “area” word is removed from all of the values present within the Type of Area column whereas the “contact” word is removed from the value within the Contact Person column.

Reason:

The “area” and “contact” word should be removed as it does not provide any useful observations with the presence of the wordings as it is already represented by the remaining wordings. The function is used as it provides a rapid process by searching for the matching wordings through the parameter and replaces it with the inputted word.

5.4 Extra Feature 4: log()

Source Code:

```
#Transform rent price to view distribution
logged_price <- log(house_data$`Rent Price`)
boxplot(logged_price ~ house_data$`City Located`)
```

Figure 5-7: log() Function

The log() function returns the natural logarithm of the value inputted, which is implemented in this scenario by passing the Rent Price column as an argument within the function and store it in the logged_price variable. A boxplot is then plotted with the logged_price according to the City Located.

Plot:

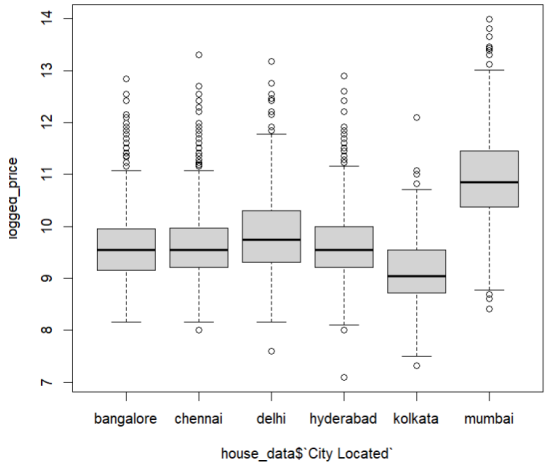

Figure 5-8: Boxplot for Logged Rent Price

Based on the boxplot, Delhi has the largest range of rent price, Kolkata has the lowest median whereas Mumbai has the highest median among all of the cities. The medians of Bangalore, Chennai and Hyderabad tend to be similar, which can also conclude on the similar rent prices present within the three cities.

Reason:

The logged values are used to provide better visualizations on the range and the median of the rent prices as the visualization is not significant enough for the normal values of the rent price.

5.5 Extra Feature 5: ggcorrplot()

Prerequisite: Install and load ggcorrplot library

Source Code:

```
#Looking into data correlation
house_corr <- select(house_data,`BHK In Unit`,`Rent Price`,`Unit Size`,`Number of Bathroom`,
                     `Price Per Square Feet`,`Total Number of Floors`)
corr <- round(cor(house_corr),1)
corrp.mat <- cor_pmat(house_corr)
ggcorrplot(corr, method ="square", hc.order = TRUE, lab=TRUE) +
  labs(title="Correlation Plot for House Data")
```

Figure 5-9: ggcorrplot() Function

The ggcorrplot() assists in constructing a correlation plot through reordering the correlation matrix and presents the level of significance on the plot so that some relationships can be determined beforehand (STHDA, n.d.). The columns needed for the correlation plot – BHK In Unit, Rent Price, Unit Size, Number of Bathroom, Price Per Square Feet and Total Number of Floors are selected with the select() function to construct a new data frame. Then, a correlation matrix is computed with the use of cor() according to the house_corr data frame, where the values present within the matrix will be rounded up to 1 decimal. Alternatively, the cor_pmat() function is also doable to compute the matrix but it will be displayed in p-values. The visualization of the correlation matrix is done through the ggcorrplot() function with hierarchical clustering by setting the hc.order argument as true. The lab argument is set a true as well to better compare the values rather than relying only on colors.

Plot:

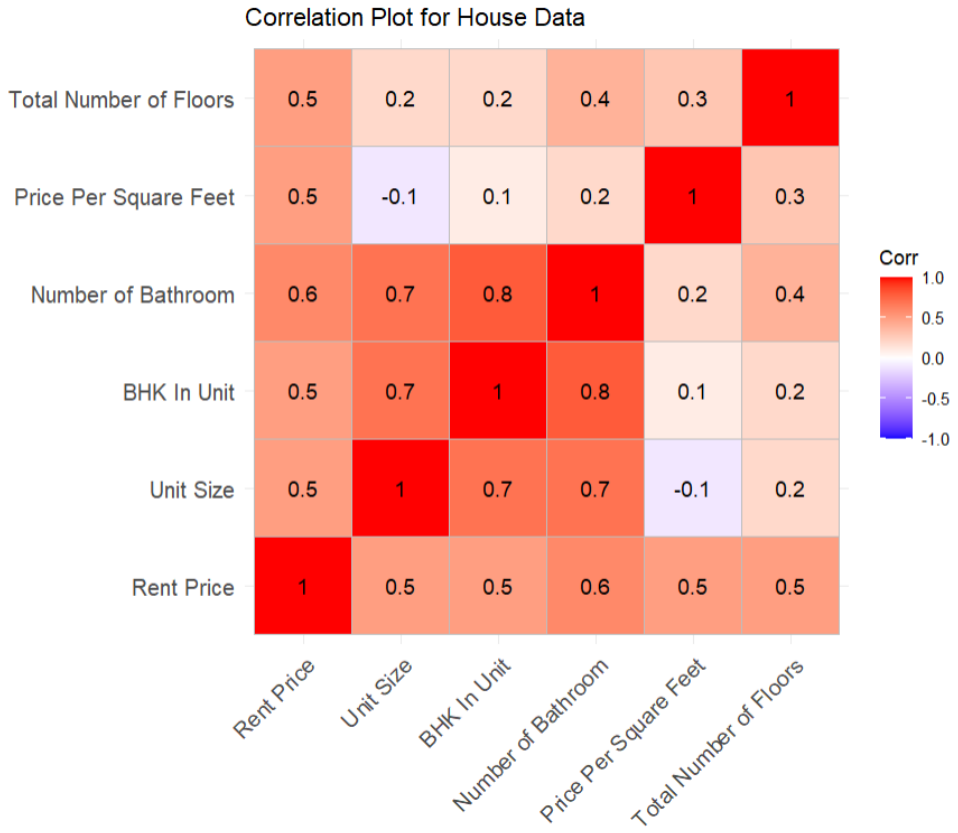

Figure 5-10: Correlation Plot

The correlation plot assists decision makers to determine the correlation between continuous variables. Most of the correlation present within the plot are positive association, with moderate positive associations mostly. The weak associations ranged from -0.1 to 0.4, which is not the majority present within the plot.

Reason:

The correlation is essential to determine the strength of associations between the variables so that accurate plots can be plotted according to the relationship between both variables.

5.6 Extra Feature 6: ntile()

Prerequisite: Install and load dplyr library

Source Code:

```
#Bin floor data
binned_data <- house_data %>% mutate("Floor Bin" = ntile(house_data$Floor, n = 5))
```

Figure 5-11: ntile() Function

The ntile() function is utilized to break up the input vector to buckets according to the value of n. The Floor Bin column is defined through assigning the Floor column into 5 bins with ntile().

Outcome:

```
> binned_data$`Floor Bin`
   [1] 1 2 2 2 2 2 1 1 2 2 2 2 2 2 2 1 2 2 1 1 2 2 3 1 1 2 2 2 1 5 1 1 1 2 1 3 2 3
  [38] 1 2 2 1 1 3 5 1 1 4 1 1 1 2 3 1 1 5 1 2 3 1 5 2 1 5 2 1 1 1 1 1 2 1 2 2 1
  [75] 1 2 1 3 1 2 1 2 1 2 2 3 1 5 2 3 1 1 3 1 2 2 2 4 2 2 1 2 1 3 1 5 1 1 1 1 1 2
 [112] 1 2 1 2 2 3 1 3 2 2 2 2 2 1 2 1 2 3 1 2 1 2 2 4 1 2 1 1 2 2 3 1 5 1 2 3 1
 [149] 2 1 1 2 1 1 1 2 1 1 2 2 2 3 1 2 3 1 1 1 3 2 1 1 1 3 2 2 1 2 3 1 1 3 2 3 2
 [186] 1 1 3 3 1 1 1 3 2 1 2 3 2 1 2 1 2 1 2 1 3 2 2 1 2 2 3 1 1 1 1 2 2 1 1 2 3
 [223] 2 2 1 1 5 1 1 2 1 1 1 1 1 1 2 1 1 2 1 1 3 2 2 2 3 3 2 2 1 2 1 2 3 1 2 3 1
 [260] 1 1 5 2 1 1 1 1 2 1 2 3 1 2 1 1 1 1 1 1 2 1 1 2 2 1 1 3 2 1 2 1 2 1 1 3 1
 [297] 2 1 1 1 1 3 1 2 2 1 1 1 1 1 1 1 1 1 3 1 1 1 2 1 1 1 4 1 1 1 1 1 2 1 2 2
 [334] 1 4 1 2 1 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 4 2 1 2 2 2 1 3 2 1 3 4 1 3 1 2 2
 [371] 1 1 3 2 1 1 2 1 2 1 1 2 2 2 1 1 1 1 2 1 2 2 2 1 1 1 2 1 2 2 2 1 2 4 1 2 1
 [408] 1 1 1 2 1 1 1 1 2 2 1 1 1 1 1 1 3 2 3 1 2 2 2 1 3 1 3 1 1 3 1 4 2 1 2 2 2
 [445] 1 1 3 1 2 1 2 1 1 1 1 1 2 1 3 2 1 2 3 2 2 3 4 1 1 1 1 1 1 1 4 1 2 1 1 1 1
 [482] 1 5 3 2 1 1 5 3 1 1 3 1 1 3 1 1 1 2 4 4 2 2 2 1 3 2 1 1 2 3 2 5 1 4 2 1 1
 [519] 1 1 1 3 2 2 3 5 5 3 5 5 3 5 3 3 3 2 3 3 5 5 3 5 4 3 3 5 3 5 5 1 3 4 2 4 5
 [556] 5 5 1 2 1 5 5 5 5 3 5 3 3 5 5 5 3 3 5 4 4 2 3 4 4 5 3 1 4 3 3 3 3 1 3 5 4 3
 [593] 3 5 3 5 5 4 5 3 5 5 5 5 5 5 3 5 4 3 5 5 5 3 5 1 5 4 3 3 5 5 3 3 5 3 3 4 5 3
 [630] 5 3 1 5 1 5 1 2 5 5 5 5 5 5 3 5 5 5 4 2 3 3 4 5 5 5 5 3 5 5 5 5 5 3 5 5 5 1
 [667] 3 3 5 5 3 3 3 5 5 5 3 5 5 5 3 5 5 2 5 5 5 5 5 5 5 3 3 3 1 5 5 5 5 3 5 3 5
 [704] 5 4 5 4 5 3 1 3 4 3 5 5 3 1 5 4 3 5 5 1 3 3 2 3 5 3 3 5 5 1 4 4 5 5 5 3 3
 [741] 5 1 3 1 5 5 5 5 5 3 3 5 5 3 5 5 5 5 2 4 5 3 5 3 3 2 5 3 4 5 3 3 5 3 5 3 3
 [778] 5 5 3 5 5 5 5 5 2 4 4 5 5 5 3 3 3 4 3 5 5 5 5 3 3 5 5 3 4 5 5 3 5 3 4 4 5 3
 [815] 5 5 3 5 5 1 5 3 3 3 5 3 3 3 3 5 5 2 2 3 5 5 5 5 4 3 3 1 3 5 3 2 5 5 3 5 5
 [852] 5 5 5 5 2 5 3 2 5 5 5 3 3 5 5 5 3 5 5 5 5 2 4 3 5 4 5 1 3 5 1 2 3 5 5 5 2 4
 [889] 4 5 5 5 3 3 5 3 3 5 5 5 3 5 3 5 4 3 4 3 4 3 5 2 4 4 5 5 5 4 3 5 2 5 5 4 3
 [926] 5 3 5 3 5 5 4 5 2 1 5 5 3 5 5 5 5 4 4 4 5 4 3 1 3 4 5 1 4 4 4 5 4 5 3 3 5
 [963] 3 3 3 3 5 3 3 5 3 3 3 3 3 5 3 1 3 4 3 3 5 2 5 3 5 5 4 5 3 5 3 5 3 3 3 5 3
[1000] 2
```

Figure 5-12: Binning for Floor Data

The floors are binned to 5 bins which can be seen in the Floor Bin column regarding the bin assigned according to its range within the data frame.

Reason:

The Floor column has to be binned for to improve the visualization as the initial data is too noisy to be interpreted or observed. Hence, through binning the floor data, the bin means can be calculated to plot a cleaner line chart rather than using a plot with noisy data.

5.7 Extra Feature 7: kmeans() and fviz_cluster()

Prerequisites: Install and load factoextra library

Source Code:

```
model <- kmeans(cluster_column, 5, nstart = 50)
print(model)
fviz_cluster(model, data = cluster_column)

#Check the count for each cluster
model$size

#Check cluster means
model$centers
```

Figure 5-13: kmeans() and fviz_cluster() Function

The kmeans() function is used to perform k-means clustering on a data matrix, the cluster_column is passed as the x argument where the clustering process should work on, 5 centers are defined through the elbow method and 50 random sets are chosen. The outcome of the model can be seen by printing it or it can also be visualized with the fviz_cluster() function, which then provides decision makers with a cluster plot (outcome explained in Analysis 2-9). The count of each cluster and the centroid values can be checked through the size and centers column within the model data frame.

Outcome:

```
Within cluster sum of squares by cluster:
[1] 150.2129 398.5458 480.5611 328.3583 468.7592
 (between_SS / total_SS =  80.8 %)
```

Figure 5-14: Cluster Sum of Squares

The between and total sum of squares is 80.8%, which is also indicating that the model is a good fit.

```
> #Check the count for each cluster
> model$size
[1]   12  165  589   75 3905
```

Figure 5-15: Count of Each Cluster

The size of each cluster indicates how many points are there within one cluster.

Cluster	Count
1	12
2	165
3	589
4	75
5	3905

Table 5-1: Count of Each Cluster

```
> #Check cluster means
> model$centers
  as.numeric.house_data..Total.Number.of.Floors.. house_data..Rent.Price.
1                                       0.8305836                11.2959868
2                                       0.5594414                 2.9362901
3                                       1.4533323                 0.3394290
4                                       5.3794338                 2.4115756
5                                      -0.3487183                -0.2562949
```

Figure 5-16: Cluster Centroids

The centroids of each cluster can assist in determining the labels given to the clusters as it provides relevant values for decision making.

Plot:

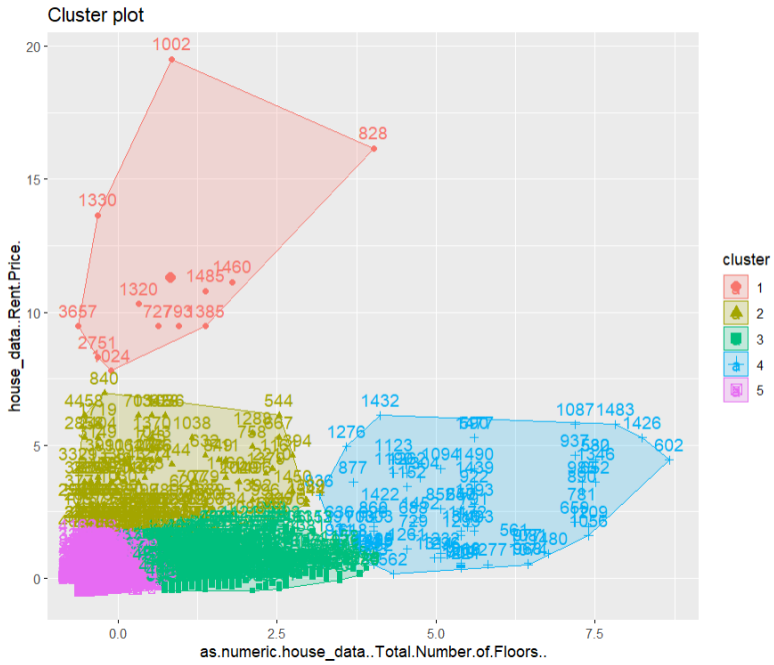

Figure 5-17: Cluster Plot

The plot is already explained in Analysis 2-9.

5.8 Extra Feature 8: str_split_fixed()

Prerequisites: Install and load stringr library

Source Code:

```
#Parse the floor number and the total floor number
house_data[c('Floor', 'Total Number of Floors')] <- str_split_fixed(house_data$`Floor Number`,' out of ', 2)
unique(house_data$Floor)
unique(house_data$`Total Number of Floors`)
```

Figure 5-18: str_split_fixed() Function

The use of str_split_fixed() is to split up a string according to a fixed number of segments. The Floor Number column from the house data is passed as the value argument, the ‘ out of ’ string will be the pattern to split up the strings by and the number of pieces to return is 2 as only 2 values are needed through the split. The split values are assigned to the new columns – Floor and Total Number of Floors respectively. The unique() function is called for each new column to determine the unique values present inside the columns.

Outcome:

Floor	Total Number of Floors
0	2
1	3
1	3
1	2
1	2
0	1
0	4

Figure 5-19: Floor and Total Number of Floors Column

The values are split and assigned into the Floor and Total Number of Floors column, which is present in the form of integers.

Reason:

The Floor Number column is parsed so that more insights can be gained through a more diverse set of values and columns as aggregations can be done to the values. Other than that, the Floor Number column is not friendly in terms of analysis as it does not support any form of calculations due to its nature of strings.

5.9 Extra Feature 9: fviz_nbclust()

Prerequisite: Install and load factoextra library

Source Code:

```
fviz_nbclust(cluster_column, kmeans, method = "wss")
```

Figure 5-20: fviz_nbclust() Function

The elbow method is selected to calculate the number of clusters present according to the cluster_column data frame. Hence, the method involves the WSS method so that the total WSS is looked into as a function of the number of clusters (Kassambara, 2017). The fviz_nbclust() function is used by passing the cluster_column data frame as the x value to be computed, the partitioning function would be in kmeans and the method to be use for determining the optimal cluster number is wss.

Plot:

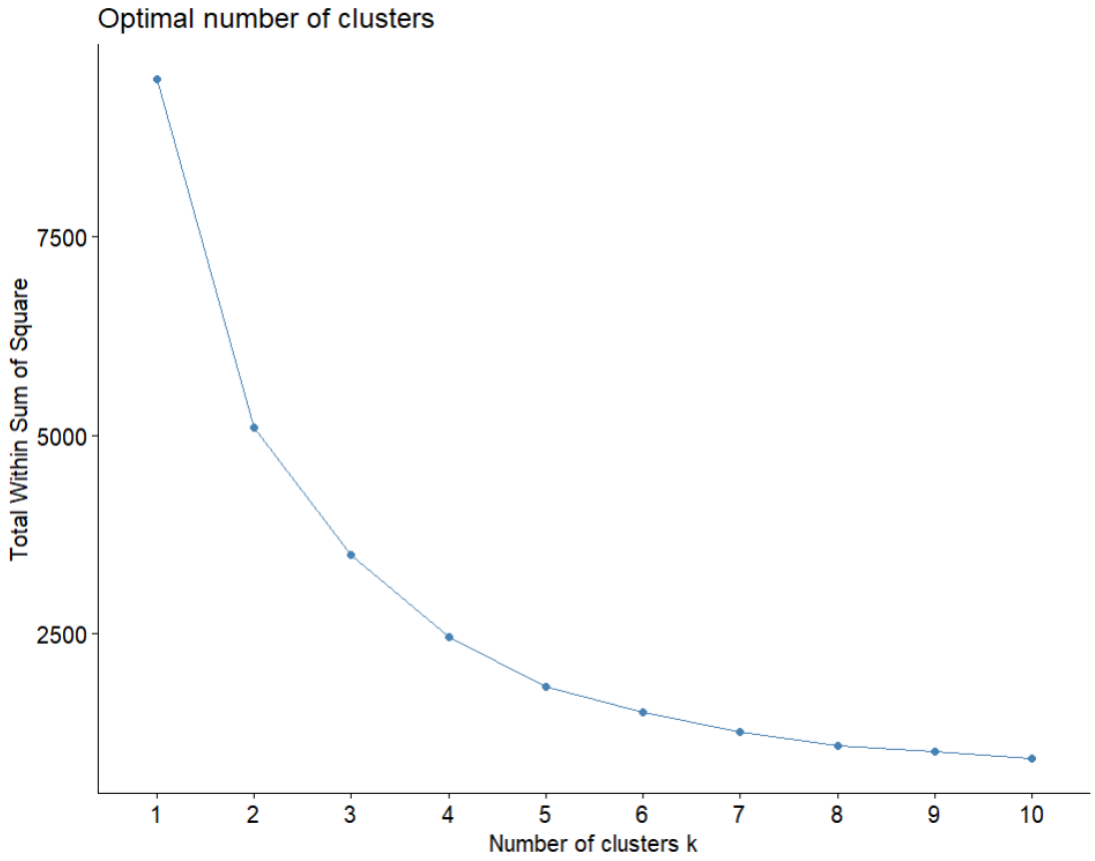

Figure 5-21: Determine the Number of Clusters

Based on the plot, 5 as the number of clusters within the solution seem to be most optimal among all as the elbow is present on 5. The elbow is determined by looking at a turning point who's shaped like an elbow on the lines, the number of clusters is determined by looking at the x-axis value of the elbow.

Reason:

The optimal number of clusters has to be determined to ensure that the data is divided in a proper and efficient manner. The appropriate number of clusters assures the granularity of clusters is proper and assists in maintaining balance between the accuracy and compressibility of the clusters.

5.10 Extra Feature 10: map_data()

Prerequisite: Install and load mapdata library

Source Code:

```
india <- map_data("world", region = "india")
```

Figure 5-22: map_data() Function

The map_data() function is used to acquire data from the maps package into data frames for plotting maps with the use of ggplot(). The name of map – world is passed as the first argument, and region-wise India is selected so that the data related to the coordinates of the India map is generated.

Outcome:

```
> india
       long      lat group order region            subregion
1  93.89004 6.831055     1     1  India        Great Nicobar
2  93.82881 6.748682     1     2  India        Great Nicobar
3  93.70928 7.000683     1     3  India        Great Nicobar
4  93.65800 7.016065     1     4  India        Great Nicobar
5  93.65635 7.136231     1     5  India        Great Nicobar
6  93.68418 7.183593     1     6  India        Great Nicobar
7  93.82246 7.236620     1     7  India        Great Nicobar
8  93.85899 7.206837     1     8  India        Great Nicobar
9  93.92959 6.973486     1     9  India        Great Nicobar
10 93.89004 6.831055     1    10  India        Great Nicobar
12 93.73360 7.356493     2    12  India       Little Nicobar
13 93.63848 7.261866     2    13  India       Little Nicobar
14 93.59727 7.318750     2    14  India       Little Nicobar
15 93.61426 7.358106     2    15  India       Little Nicobar
16 93.65469 7.379932     2    16  India       Little Nicobar
17 93.69247 7.410595     2    17  India       Little Nicobar
18 93.73360 7.356493     2    18  India       Little Nicobar
20 93.44257 7.877832     3    20  India      Katchall Island
21 93.36504 7.876562     3    21  India      Katchall Island
22 93.34200 7.919336     3    22  India      Katchall Island
23 93.30937 7.964014     3    23  India      Katchall Island
24 93.33447 8.006933     3    24  India      Katchall Island
25 93.37549 8.017920     3    25  India      Katchall Island
26 93.43369 7.948389     3    26  India      Katchall Island
27 93.44736 7.899121     3    27  India      Katchall Island
28 93.44257 7.877832     3    28  India      Katchall Island
30 93.53691 8.056641     4    30  India       Trinkat Island
31 93.49004 8.019434     4    31  India       Trinkat Island
32 93.47822 8.024463     4    32  India       Trinkat Island
33 93.47177 8.052686     4    33  India       Trinkat Island
34 93.46973 8.072656     4    34  India       Trinkat Island
35 93.46123 8.108594     4    35  India       Trinkat Island
36 93.45644 8.171875     4    36  India       Trinkat Island
37 93.49404 8.224659     4    37  India       Trinkat Island
38 93.53164 8.213770     4    38  India       Trinkat Island
39 93.51162 8.159766     4    39  India       Trinkat Island
40 93.53691 8.056641     4    40  India       Trinkat Island
42 73.06738 8.269093     5    42  India       Minicoy Island
43 73.05332 8.256689     5    43  India       Minicoy Island
44 73.03886 8.251953     5    44  India       Minicoy Island
45 73.02852 8.253515     5    45  India       Minicoy Island
```

Figure 5-23: India Data

The data frame generated from the map_data() function consists of the longitude, latitude, group, order, region and subregion of the Indian map, where the data is then used to illustrate the shape of the country with the help of geom_polygon().

Reason:

The data relating to the coordinates of each cape, shore and inland has to be captured so that the coordinates can be used to draw out a map of India to determine the geographical locations of each city. The map can be drawn according to accurate prove of coordinates so that errors can be avoided. Hence, improving the presentation of data and assists in extracting more information.

5.11 Extra Feature 11: ggdotchart()

Prerequisites: Install and load ggpubr library

Source Code:

```
ggdotchart(fam_area, x = "area_label", y = "area_type_fam",
           color="area_label",
           palette = c("#00AFBB", "#E7B800", "#FC4E07"),
           sorting = "descending",
           rotate = TRUE,
           y.text.col = TRUE,
           dot.size = 15,
           label = area_type_fam,
           font.label = list(color = "white", size = 9,
                             vjust = 0.5),
           ggtheme = theme_pubr()
)+
  theme_cleveland() +
  ggtitle("Distribution of The Area Types for Families") +
  ylab("Number of Families")
```

Figure 5-24: Plotting Cleveland's Dot Plot

The Cleveland's dot plot is an alternative to bar charts so that more diverse types of visualizations are present rather than having only bar plots and pie charts present as charts for analysis. To plot the dot plot, the gggotchart() function is called by passing the fam_area data frame as the data, area_label as the x variable and group so that the grouping on elements of the x variable can be indicated and area_type_fam as the y variable. The palette argument is passed with 3 colors in the form of hex codes as there is only 3 types of data present in the fam_area data frame. The character vectors are sorted according to the descending order, by implementing rotate, the plot is rotated to a horizontal plot orientation, y_text_col is set as true so that the y axis texts are colored according to groups. The dot.size argument is to adject the size of the dots present in the dot plot, the size is set to 15 so that the labels can be fitted in. The label consists of area_type_fam as the name of the column that contains the labels of points. Other than that, the font.label argument passes a list which consists of the combination of elements such as white as color, 9 as size and 0.5 for vjust to determine the aesthetics of the label. The theme for the dot plot is then set to theme_pubr(), whereas theme_cleveland() is called to add the dotted lines for the plot. The plot is named as Distribution of the Area Types for Families as the title and Number of Families as the y-axis label.

Plot:

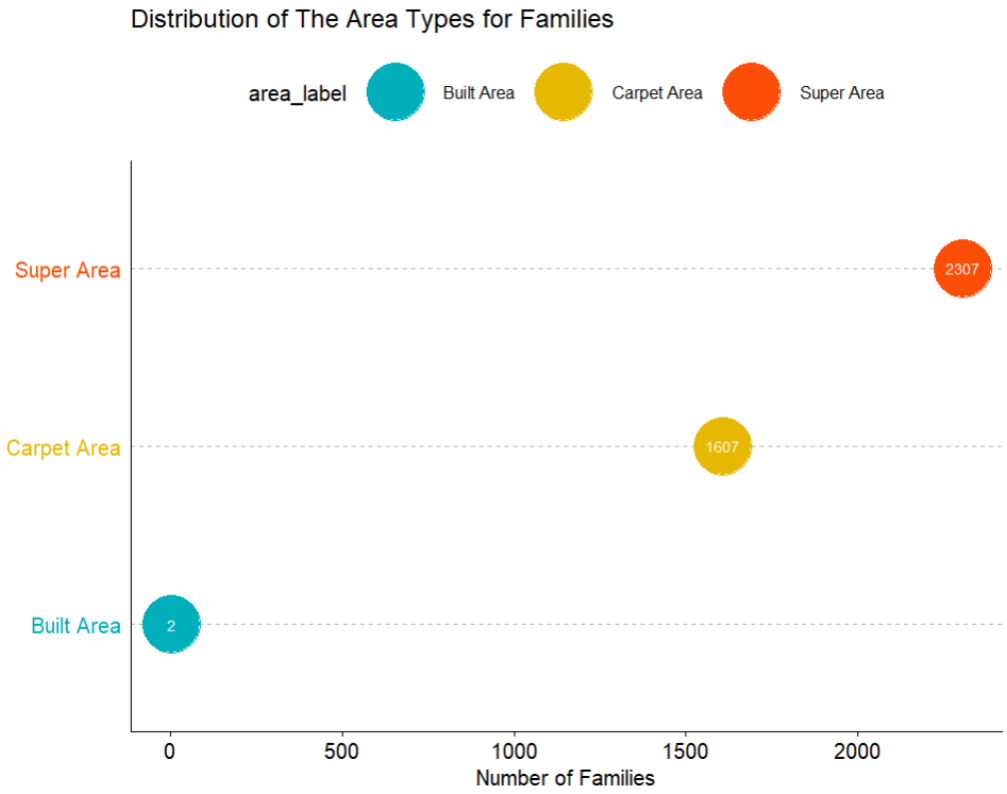

Figure 5-25: Cleveland's Dot Plot

Different values in the Cleveland's dot plot can be determined through the position of the point along a common scale and the label present on the points. As the point moves towards the right side, the higher the value is and vice versa. The groupings of the data can also be varied through the use of color labels on the points.

Reason:

Other than bar plots, other alternatives are encouraged to be explored so that more options are available while plotting charts according to categorical values. The use of dots instead of bars reduces redundancy and clutter. Hence, it is easier for decision makers to read and make comparison between the differences between the values.

5.12 Extra Feature 12: Donut Plot

Prerequisites: Install and load ggplot2 library

Source Code:

```
# Compute percentages
fam_floor$fraction = fam_floor$floor_fam / sum(fam_floor$floor_fam)

# Compute the cumulative percentages (top of each rectangle)
fam_floor$ymax <- cumsum(fam_floor$fraction)

# Compute the bottom of each rectangle
fam_floor$ymin <- c(0, head(fam_floor$ymax, n=-1))

# Compute label position
fam_floor$labelPosition <- (fam_floor$ymax + fam_floor$ymin) / 2

# Compute a good label
fam_floor$label <- paste0(fam_floor$label_fam, "\n value:", fam_floor$floor_fam)

ggplot(fam_floor, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=label_fam)) +
  geom_rect() +
  geom_label( x=3.5, aes(y=labelPosition, label=label), size=3) +
  scale_fill_brewer(palette=3) +
  coord_polar(theta="y") +
  xlim(c(2, 4)) +
  theme_void() +
  theme(legend.position = "none") +
  ggtitle("Distribution of the Preference of Floors Among Families")
```

Figure 5-26: Plotting Donut Plot

Donut charts is used as an alternative for pie charts as it is visualized as a ring divided into sections that represents a proportion of the whole ring. To plot the donut chart, the fraction of each category, the cumulative percentages, bottom of the rectangle, label position and the label value should be determined beforehand. For example, the chart is visualized with ggplot() where fam_floor is the data, the cumulative percentage is the ymax value, the bottom of the rectangle as ymin, 4 as the xmax and 3 as the xmin value to maintain the shape of the donut. The geom_rect() function is utilized to plot each groupings as a rectangle, coord_polar() to convert the stacked rectangles into a ring shape and xlim() to change a pie chart to a donut by adding an empty circle in the middle.

Plot:

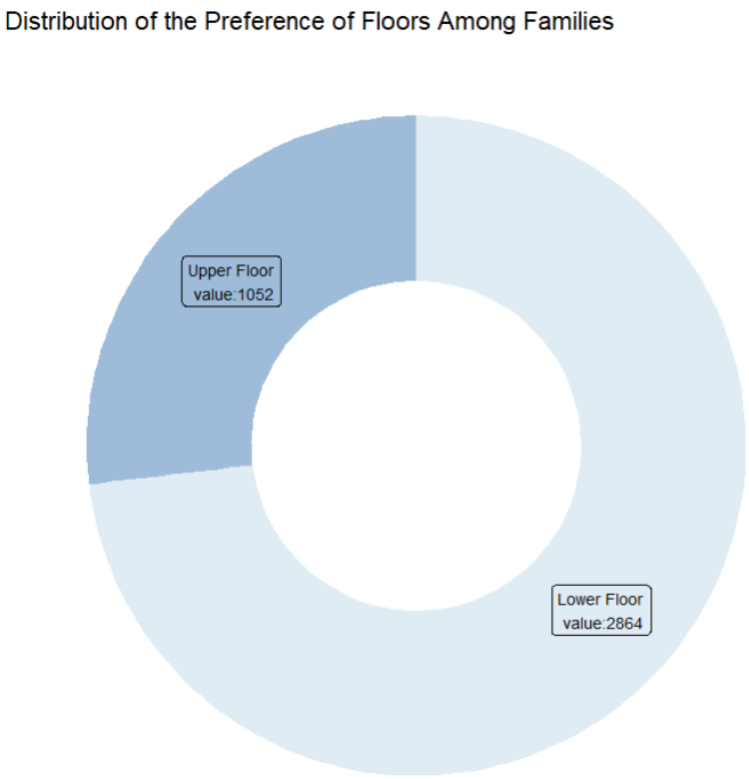

Figure 5-27: Donut Plot

The donut plot displays the sections according to the percentage of each value present within the categories. The category who obtains a larger portion of the donut are larger values whereas smaller portion indicates smaller values.

Reason:

The visualizations can be improved by providing more varieties of charts that can maintain the engagement of decision makers instead of looking at charts with the same concept and design and it is user-friendly as the donut chart holds a high level of similarity with the classic pie chart.

5.13 Extra Feature 13: Radial Plot

Prerequisites: Install and load ggplot2 library

Source Code:

```
ggplot(bathroom_fam_tab) +
  geom_hline(
    aes(yintercept = y),
    data.frame(y = c(0:1) * 1000),
    color= "lightgrey") +
  geom_col(aes(
    x = reorder(str_wrap(bathroom_label_fam, 8), bathroom_number_fam),
    y = bathroom_number_fam,
    fill = bathroom_label_fam),
    position = "dodge2",
    show.legend = TRUE,
    alpha= .9) +
  geom_segment(aes(
    x = reorder(str_wrap(bathroom_label_fam, 8), bathroom_number_fam),
    y = 0,
    xend = reorder(str_wrap(bathroom_label_fam, 8), bathroom_number_fam),
    yend = 2000),
    linetype = "dashed",
    color = "gray12") +
  coord_polar() +
  labs(title = "Number of Bathrooms Preferred By Families", ylab = "Count",
       xlab="Number of Bathrooms")
```

Figure 5-28: Plotting Radial Plot

Radial plots can be considered as another form for alternatives of bar charts. With the help of geom_hline(), a custom panel grid is made by determining the y-intercept, data frame value and the color. Then, geom_col() is added to add bars that represent the cumulative number of bachelors, str_wrap() wraps the text so that each line has maximum 8 characters. However, it does not break long words. The use of geom_segment() generates the lollipop shaft for the number of bachelors per number of bathroom and lastly coord_polar() is used to make the graph appear in a circular manner. The title of the plot is places as – Number of Bathrooms Preferred by Families.

Plot:

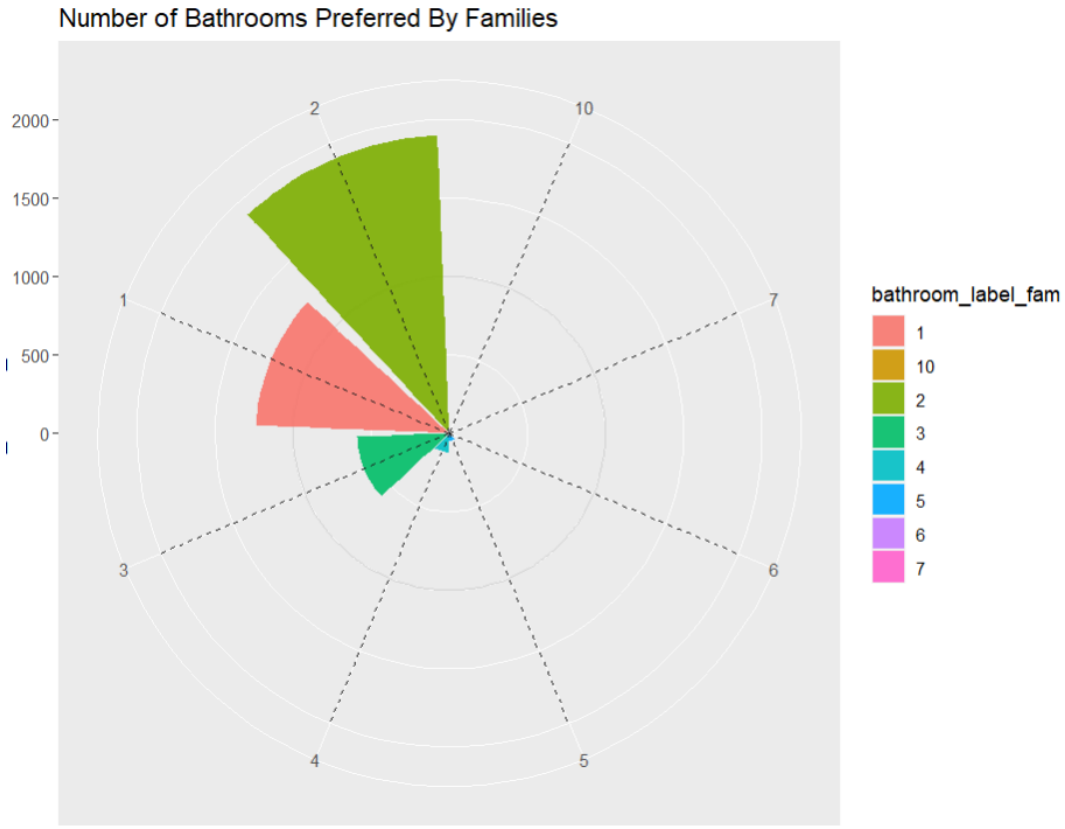

Figure 5-29: Radial Plot

The radial plot appears by having each bar displayed along the circle in the coned shape instead of implementing a line. The size of the coned bar varies according to the number of bachelors present for this graph, the higher the number of bachelors, the bigger the cone size and vice versa. The plot arranges itself in a clockwise manner according to the number of bachelors to ease visualization and each category is represented by their respective colors.

Reason:

A radial plot is considered as another alternative to replace the bar plot in terms of plotting the count of categorical or continuous values. The variation of cones has also ease decision makers in terms of visual as the size of the cones properly represents the value of the cone. Other than that, better comparisons can be done between the categories due to the obvious size difference between the cones.

5.14 Extra Feature 14: Tree Map

Prerequisites: Install and load ggplot2 and treemapify library

Source Code:

```
ggplot(city_june_dist, aes(area = n, fill = n, label = paste(`City Located`,
                                                             labels, sep = "\n"))) +
  geom_treemap() +
  geom_treemap_text(colour = "white",
                    place = "centre",
                    size = 15) +
  scale_fill_gradient(low = "orange", high = "purple") +
  ggtitle("Demand of Properties for Each City In June 2022") +
  labs(fill = "Total Properties")
```

Figure 5-30: Plot Tree Map

The tree map serves as an alternative for the pie chart, in this scenario, it can be plotted by passing the city_june_dist as data, n as area and fill, the city located, and labels are formatted to be the label. The geom_treemap() function is called to construct the tree map and geom_treemap_text() is called to add the labels on the grids to visualize the distribution of percentages of values represented by the grid. In terms of aesthetics, scale_fill_gradient(), ggtitle() and labs() are called to enhance the outlook and the user-friendliness of the chart.

Plot:

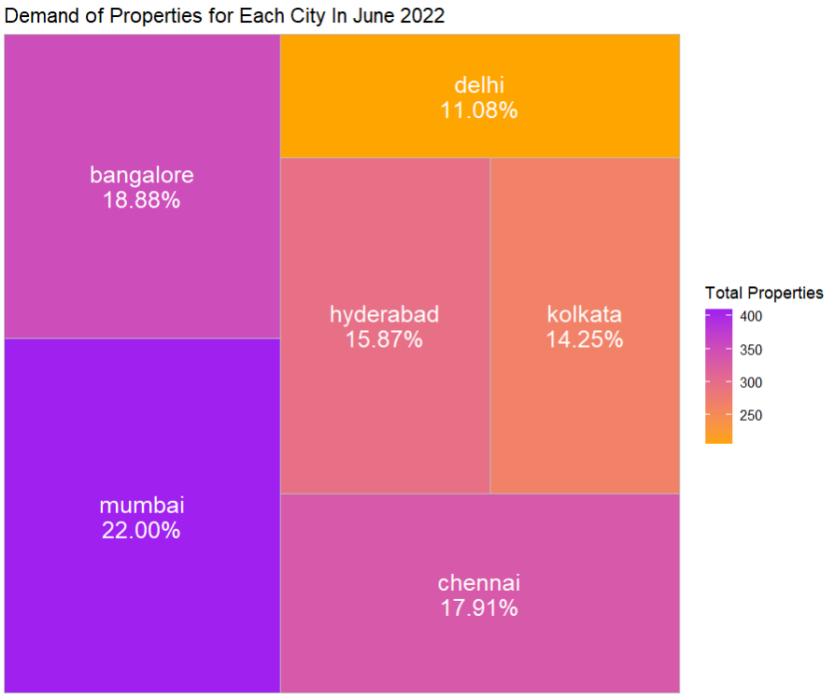

Figure 5-31: Tree Map

The tree map provides decision makers with a hierarchical view of the data present by splitting up the rectangles based on the quantitative variable, where it can be differentiated by the different sizes and orders. For instance, the biggest rectangle in the tree map indicates that the value present within the category is the largest and vice versa.

Reason:

The tree map is used as an alternative for the pie chart because they allow a rapid perception of the large contributors to the overall picture. Colors can determine the performance of the items in terms of underperforming or overperforming compared to the other elements from the same category.

6.0 Conclusion

To conclude, a few questions relating to the problem statement from the perspective of tenants and house owners are raised. Hence, to answer to the questions, a few analyses were done to determine the solutions to the questions, with well-presented graphs and values. The preference of bachelors and families in selecting their ideal house for rental is influenced by identical factors such as BHK In Unit, Unit Size, Furnishings, Rent Price, Number of Bathrooms, Floor and the Total Number of Floors. However, the analysis has also shown that both parties have similar preferences despite having different marital status and lifestyle, families only require bigger unit sizes to fit in more family members and a higher rental price as they might have a more stable income compared to bachelors. The rent price itself is also affected by common factors, but the most common relationship found within the factors are the pricing of the house increases when the unit size is bigger and contains furnishings, which usually people do not prefer. Through the analysis, the influence of geographical locations towards the total number of floors for each building is explored and the results state that the geographical characteristics of the building located might not bring a giant impact, but it might be affected by the populations and the income of the city. Lastly, the demand and pricing of the houses are also affected by the timing as different level of demands changes according to the months and the rent prices tend to vary according to the demands.

7.0 References

Ahlfeldt, G., & Barr, J. (2020, July 28). *Do skyscrapers make economic sense?* Retrieved from The London School of Economics and Political Science: https://blogs.lse.ac.uk/businessreview/2020/07/28/do-skyscrapers-make-economic-sense/

Bloomingrock. (n.d.). *7 Reasons Why High-Rises Kill Livability*. Retrieved from Smartcitiesdive: https://www.smartcitiesdive.com/ex/sustainablecitiescollective/7-reasons-why-high-rises-kill-livability/561536/

Datanovia. (n.d.). *K-Means Clustering in R: Algorithm and Practical Examples*. Retrieved from Datanovia: https://www.datanovia.com/en/lessons/k-means-clustering-in-r-algorith-and-practical-examples/

dplyr. (n.d.). *Count observations by group*. Retrieved from dplyr: https://dplyr.tidyverse.org/reference/count.html

GeeksforGeeks. (2021, December 20). *How to Use Par Function in R?* Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/how-to-use-par-function-in-r

ggplot2. (n.d.). *Gradient colour scales*. Retrieved from ggplot2: https://ggplot2.tidyverse.org/reference/scale_gradient.html

Houseey Tips. (2020, May 7). *6 Reasons to Settle in Bangalore*. Retrieved from Houseey Tips: https://www.houseey.com/blog/6-reasons-to-settle-in-bangalore

Johnson, D. (2022, September 17). *Factor in R: Categorical Variable & Continuous Variables*. Retrieved from Guru99: https://www.guru99.com/r-factor-categorical-continuous.html

Johnson, M. (2017, November 9). *Buying a Home at the End of the Year Can Really Pay Off (in Actual Dollars) — Here's How*. Retrieved from Apartment Therapy: https://www.apartmenttherapy.com/save-money-by-house-shopping-at-the-end-of-the-year-252348

Kassambara. (2017, September 7). *Determining The Optimal Number of Clusters: 3 Must Know Methods*. Retrieved from R & Data Mining: http://www.sthda.com/english/articles/29-cluster-validation-essentials/96-determiningthe-optimal-number-of-clusters-3-must-know-methods/

Mishra, S. (2022, Novermber 1). *Carpet area, built up area and super built up area: Know the difference*. Retrieved from Housing.com: https://housing.com/news/real-estate-basics-part-1-carpet-area-built-up-area-super-built-up-area

Nguyen, L. (2021, December 29). *EDA, Data Preprocessing, Feature Engineering: We are different!* Retrieved from Medium: https://medium.com/@ndleah/eda-data-preprocessing-feature-engineering-we-are-different-d2a5fa09f527

Placify. (2022, April 28). *Mumbai VS Kolkata City Comparison | Kolkata VS Mumbai Comparison*. Retrieved from Placify: https://placify.in/mumbai-vs-kolkata-city-comparison/

Pratt, M. K. (n.d.). *Data Transformation*. Retrieved from SearchDataManagement: https://www.techtarget.com/searchdatamanagement/definition/data-transformation

STHDA. (n.d.). *ggcorrplot: Visualization of a correlation matrix using ggplot2*. Retrieved from STHDA: http://www.sthda.com/english/wiki/ggcorrplot-visualization-of-a-correlation-matrix-using-ggplot2'

Tableau. (n.d.). *Guide To Data Cleaning: Definition, Benefits, Components, And How To Clean Your Data*. Retrieved from Tableau: https://www.tableau.com/learn/articles/what-is-data-cleaning#advantages-benefits